THIAGO MELLO

NARIZ - UM SISTEMA DE CORRELACIONAMENTO DISTRIBUÍDO DE ALERTAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Roberto A. Hexsel

CURITIBA

2004

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno *Thiago Eugênio Bezerra de Mello*, avaliamos o trabalho intitulado, "*NARIZ - SISTEMA DE CORRELACIONAMENTO DISTRIBUÍDO DE ALERTAS*", cuja defesa foi realizada no dia 22 de junho de 2004, às nove e trinta horas, no Auditório do Departamento de Informática do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 04 de junho de 2004.

Prof. Dr. Roberto André Hexsel
DINF/UFPR – Orientador

Prof. Dr. Carlos Alberto Maziero
PUC/PR - Membro Externo

Prof. Dr. Elias Procópio Duarte Jr
DINF/UFPR – Membro Interno

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

Dedico esta dissertação de mestrado aos meus pais, Eugênio Mello e Ivana Mello.

Agradecimentos

Agradeço a meus Pais, Eugênio Ramos Bezerra de Mello e Ivana Suely Paiva Bezerra de Mello, e ao meu irmão Bruno Bezerra de Mello, pelo apoio e incentivo. Agradeço também ao restante da minha família que sempre me apoiou. À minha namorada Renata Duarte pela paciência, compreensão e apoio. Agradeço ao Professor e ex-orientador Elias Procópio Duarte Jr. e ao Professor Carlos Maziero, pelas valiosas considerações. Ao Professor e ex-orientador Martin A. Musicante pelo período de orientação e apoio.

Agradeço ao meu orientador, Professor. Roberto A. Hexsel, por me orientar, pelos conselhos e pela amizade no decorrer da orientação.

Aos meus amigos do LARSIS, Andreas, Egon, Patricia e Bona, e do mestrado Araujo, Tiaguinho, João Eugênio, Evandro e Gustavo, pelas as ajudas, festas e conversas no café. Aos meus amigos do LEG, Professor Paulo Justiniano, Eduardo Sant'Ana, Marcos Aurélio e Pedro Ribeiro. Ao Departamento de Informática da UFPR, especialmente aos Professores Alexandre Direne e Marcos Castilho, por proporcionar o suporte institucional e incentivo.

Um agradecimento especial a Shelley, por tantas horas de trabalho.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Resumo

Com o aumento das taxas transmissão de dados em redes, novos tipos de ataques e suas quantidades têm aumentado. Sistemas de detecção de intrusão são ferramentas essenciais para a segurança de redes de computadores. Esses sistemas, quando implementados em redes com altas taxas de tráfego e ataques, processam muitas informações e podem gerar grandes volumes de evidências de tentativas de ataques através de alertas. Torna-se portanto necessário um sistema que produza, de forma resumida, evidências para análise por um humano. Esse trabalho descreve um sistema de correlacionamento distribuído de alertas, chamado Nariz. O Nariz baseia-se em duas fases de correlacionamento, com pré-processamento local e pós-processamento distribuído. O correlacionamento distribuído de alertas é uma técnica nova de correlacionamento de alertas. O sistema Nariz visa correlacionar alertas de forma distribuída em uma rede de alta velocidade, através de sub-sistemas de correlacionamento que podem ser executados em computadores com custo menor do que em sistemas centralizados. O correlacionamento distribuído utiliza troca de mensagens entre seus correlacionadores, que estão espalhados pela rede. O alerta é encaminhado ao administrador da rede quando o sistema tem vários indícios sobre uma tentativa de ataque. Mostramos em resultados experimentais que com esse mecanismo o Nariz pode reduzir o número de alertas sobre um mesmo evento, bem como o número de falsos positivos.

Abstract

The ever increasing network transmission rates give rise to new forms of attack as well as to an increase in their frequency. Intrusion detection systems (IDS) are essencial to computer network security. When these systems operate in networks with high traffic densities, and therefore under frequent attacks, they must process a big amount of information and thus may produce large number of reports of intrusion attempts. It is important that an IDS generate reports in a volume that are suitable for analysis by a human being.

A distributed alert correlation system, named Nariz (nose) is described. This system is composed by several instances of alert correlators, and these correlate intrusion alerts in two steps, first locally and then accross all Narizes.

The distibuted correlation of alerts is a new technique that is well suited to high speed networks since it can be implemented on machines with lower computing power than it would be possible with centralized correlation. Lower power machines are normally also lower cost machines.

The individual correlators exchange messages to correlate intrusion alerts and a message is sent to a human manager whenever the distributed system collects enough information regarding an intrusion attempt. In this way, we show through experimental results that the Nariz system can reduce the number of messages sent to the human overseers, while eliminating some of the false positives.

Capítulo 1

Introdução

O avanço da tecnologia de redes de computadores proporciona maiores velocidades de conexão e aumento da interconectividade de redes. Essas duas tendências acarretaram também novos problemas para projetistas de redes de computadores, incluindo má configuração de ambientes interconectados, erros de programação de aplicativos de redes, capacidade técnica limitada de administradores de redes e erro na especificação de protocolos de comunicação. Todos esses problemas resultam em constantes ameaças virtuais. Essas ameaças se manifestam através de vários tipos de ataques a redes de computadores, tais como vírus de computadores, aproveitamento de falhas de protocolos/aplicativos e sobrecarga em sistemas computacionais, sendo que estes correspondem a uma parcela pequena dos ataques que atualmente ocorrem na Internet [1].

Para reagir a essas ameaças foram desenvolvidos duas abordagens, uma abordagem proativa e uma abordagem reativa [38]. A abordagem proativa é aplicado através de *políticas de segurança* para tentar minimizar as ameaças às redes de computadores [30]. A abordagem reativa é aplicado depois que a política de segurança é violada, sendo conhecido como *resposta a incidente*. Tanto a política de segurança, como a resposta a incidente, são abordagens que devem ser integrados para haver trocas de informações constantes entre estes, minimizando desta forma as conseqüências de ataques às redes de computadores.

Este capítulo está organizado da seguinte forma. Primeiro é apresentada uma breve introdução de sistemas de detecção de intrusão e a arquitetura empregada para detecção de intrusão em redes de alta velocidade. Em seguida, discute-se o correlacionamento de alertas providos de sistemas de detecção, e também o problema da quantidade de alertas em redes de alta velocidade e como este pode ser resolvido. Então, é apresentado um sistema de correlacionamento de alertas para detecção de intrusão distribuída em uma rede de alta velocidade.

1.1 Sistemas de Detecção de Intrusão

Sistemas de Detecção de Intrusão vêm sendo muito utilizados em grandes organizações como uma das abordagem proativas de segurança de redes. Esses sistemas são essenciais em políticas de segurança, e colaboram para a resposta a incidentes através de *alertas* emitidos pela detecção das tentativas de ataques. Os alertas possuem informações sobre a tentativa de ataque e são emitidos por Sistemas de Detecção de Intrusão quando estes suspeitam de alguma tentativa de ataque. Essas suspeitas são geradas a partir de informações obtidas de ações dos atacantes [15].

Existem dois tipos principais de Sistemas de Detecção de Intrusão [14]: sistemas que detectam tentativas de ataques através de ações suspeitas em uma rede, e sistemas que detectam tentativas de ataques através de ações em um máquina (*host*). Existem variações nesses tipos de sistemas de detecção de intrusão, bem como combinações desses dois tipos de sistemas. Nos sistemas de detecção podem ser usadas três técnicas capazes de identificar ações que revelam ser tentativas de ataques, que são a (1) técnica de detecção de intrusão baseada em ações conhecidas de ataques, (2) detecção de intrusão através de heurísticas e cálculo de probabilidade, e (3) detecção de intrusão híbrida, baseada nas duas técnicas anteriores.

Redes de alta velocidade estão se tornando cada vez mais comuns. Sistemas de detecção de intrusão detectam atividades hostis através de evidências ou suspeitas de ataques. Em uma rede de alta velocidade os Sistemas de Detecção de Intrusão podem se tornar ineficazes por causa do elevado tráfego que os sistemas de detecção têm que

processar nessa rede. A alternativa que pode ser adotada para contornar esse problema é distribuir o tráfego para sistemas de detecção de intrusão distribuídos pela rede. Em cada um dos sub-sistemas é analisada uma parcela do tráfego da rede, podendo ocorrer que mais de um sub-sistema de detecção descubra a mesma tentativa de ataque. Essa abordagem de detecção de intrusão distribuída pode detectar uma grande quantidade de tentativas de ataque, gerando assim maior quantidade de alertas sobre essas tentativas. É possível melhorar a qualidade dos alertas através de processamento mais sofisticado dos alertas recebidos.

1.2 Correlacionamento de Alertas

Quanto mais rápida a conexão de rede, possivelmente mais tentativas de ataques em sistemas de detecção distribuídos são evidenciadas, e portanto mais alertas são gerados. Quando os alertas são gerados por sistemas de detecção, o administrador deve ser capaz de analisar os alertas e extrair deles os comportamentos que ameaçam a segurança da rede.

Quando há uma grande quantidade de alertas, é necessário que um sistema os relacione mais rapidamente do que é possível com a análise manual dos alertas. Estes sistemas se baseiam no *correlacionamento de alertas* provindos de sistemas de detecção de intrusão, com o intuito de minimizar a quantidade de alertas que devem ser analisados. Em uma rede de alta velocidade o processamento dos alertas pode ser mais custoso devido à quantidade de alertas que os sistemas de detecção podem emitir por causa do elevado tráfego.

1.3 Sistema de Correlacionamento Distribuído de Alertas

Para detectar as ameaças às redes, o sistema de detecção de intrusão utilizado neste trabalho é o Snort [59, 50]. Para relacionar os alertas é utilizado um sistema de correlacionamento de alertas distribuído, chamado Nariz, baseado em uma arquitetura de detecção de intrusão distribuída.

O Nariz utiliza alertas estruturados de suspeitas de ataques para relacioná-las na busca de suspeitas mais concretas. Esse relacionamento é executado por dois módulos: um que relaciona os alertas localmente e outro que distribui tais alertas, para um relacionamento posterior. A concretização do correlacionamento de alertas consiste da avaliação de regras, onde cada regra é avaliada para auxiliar o julgamento de uma suspeita, diminuindo desta forma o número de alertas e o número de alertas de suspeitas infundadas.

1.4 Organização do Trabalho

O restante deste trabalho está organizado da seguinte maneira. O capítulo 2 apresenta conceitos mais detalhados de sistemas de detecção de intrusão, como seu histórico, tipos de sistemas de detecção e como se detectam as tentativas de ataques. O capítulo 3 descreve a implementação de um sistema de detecção de intrusão com desempenho otimizado. Também são apresentados resultados de experimentos realizados com sistemas de detecção em redes de alta velocidade. O capítulo 4 apresenta o sistema desenvolvido nesse trabalho de correlacionamento distribuído de alertas. As conclusões seguem no capítulo 5. No Apêndice A está descrito o algoritmo utilizado para correlacionamento distribuído de alertas, e o Apêndice B contém os resultados detalhados dos experimentos descritos no capítulo 4.

Capítulo 2

Sistemas de Detecção de Intrusão

Com o aumento da popularidade da Internet, tem aumentado a incidência de ataques explorando vulnerabilidades em sistemas computacionais conectados [71], e todo sistema computacional conectado a uma rede é um alvo em potencial para atacantes [38]. As possibilidades de ataques são: (1) acesso não autorizado a informações ou serviços, (2) manipulação não autorizada dessas informações ou serviços, e (3) meios que tornem essas informações ou serviços indisponíveis a seus usuários. Como existem diversas formas de ataques, é imprescindível a utilização de um mecanismo de prevenção dessas ameaças. Contudo, o uso desses mecanismos não previne todos os ataques. Existem diversos mecanismos para maximizar a prevenção e a detecção de ataques, que são discutidos em [8, 46, 49] e os mais listados a seguir:

***Firewall*:** um *firewall* é uma ferramenta que aplica políticas/regras de segurança ao tráfego entre uma rede confiável e uma rede não confiável [51]. Essas regras controlam o tráfego que passa entre essas duas redes, negando-o ou aceitando-o, conforme as regras adotadas [24].

Análise de *logs*: *logs* são arquivos que mantém registro de eventos que ocorrem em uma máquina (*host*). A análise de *logs* é executada através de ferramentas ou procedimentos que permitem uma análise próxima ao tempo-real dos eventos ocorridos no sistema.

***Patch*:** um *patch* é um trecho de código que visa corrigir um aplicativo que possui alguma vulnerabilidade. Esse mecanismo pode ser eficiente em aplicativos ou em sistemas operacionais se houver a freqüente atualização do sistema por *patch*es, mas essa atualização só é realizada e conhecida depois que a vulnerabilidade é descoberta.

Cifragem de dados: Cifragem de dados é uma técnica que utiliza a criptografia para "embaralhar" os dados de modo que intrusos não possam decifrá-los [31].

Lista de controle de acesso: uma lista de controle de acesso é um conjunto de regras que é usada para permitir ou negar certas atividades, acessos ou conexões em uma rede. Lista de controle de acesso pode ser implementado, por exemplo, em do *firewall* ou em um roteador,

Os *firewalls*, em sua maioria, têm duas desvantagens: (1) usar regras estáticas, e (2) não possuírem descrição (informações) das tentativas de acesso não autorizados que foram bloqueadas [49]. Isso quer dizer que, caso seja negado algum acesso, o *firewall* não tem informações suficientes para julgar se foi ou não uma tentativa de ataque. Outro problema é que quando o acesso é autorizado, mas com intenções de ataque, o *firewall* não examina o resultado desse acesso.

A análise de *logs* torna-se ineficaz quando há uma grande quantidade de registros nos arquivos de registro, pois o longo tempo de análise pode tornar o mecanismo inviável [49]. Além dessa ineficiência, a análise de *logs* pode prover somente o estado em que o sistema se encontra.

Há constantes descobertas de vulnerabilidades em sistemas computacionais. Torna-se com isso necessário um mecanismo de reconhecimento dessas vulnerabilidades e correção de forma automatizada através de *patch*es.

Apesar de o mecanismo de encriptação de dados prover uma maior segurança dos dados e dos recursos computacionais, esse mecanismo pode falhar diante de três situações: (1) ataques em que não há interesse no acesso aos dados, e sim na indisponibilidade dos dados; (2) ataques para a quebra de senhas criptografadas; (3) ataques aos dados quando forem descriptografados, ou antes de serem criptografados. As listas de controle de acesso têm

os mesmos problemas que as regras estáticas apresentam nos *firewalls*.

Enfim, não há mecanismo que forneça total segurança a um sistema computacional. Existe então a necessidade de desenvolver meios que detectem, analisem e reajam de forma adequada, ao maior número de tentativas de ataques. Esses meios de prevenção de ataques são chamados de Sistemas de Detecção de Intrusão (SDIs, ou *Intrusion Detection Systems*) [65].

SDIs são ferramentas automatizadas para detecção de intrusão que podem ser implementados em software ou hardware [54]. Sua finalidade é detectar atividades incomuns, impróprias ou anômalas, e conseqüentemente, detectar ataques. Um SDI procura identificar *evidências de ataques* e essas evidências podem caracterizar um ataque, ou um conjunto de ataques. Se não houverem informações suficientes, ou se as informações não forem confiáveis, o SDI não consegue concluir se as evidências disponíveis indicam a ocorrência de um ataque [46].

Um SDI pode analisar o tráfego de entrada e de saída de uma rede, ou dados locais de uma máquina na qual é executado, à procura de tentativas de ataques que têm diversas naturezas.

Os Sistemas de Detecção de Intrusão permitem que organizações se previnam contra as ameaças que o aumento da conectividade entre redes provê e também proporcionam um aumento na confiança nas informações trocadas via rede. Em várias organizações os sistemas de detecção de intrusão estão tornando-se mecanismos essenciais em suas infra-estruturas de segurança.

Há diversas razões para utilizar Sistema de Detecção de Intrusão para aumentar a segurança, incluindo: (1) aumentar a capacidade de percepção de riscos de ataques e a descoberta de ataques, (2) detectar erros que administradores e usuários do sistema podem cometer, e (3) a identificação de vulnerabilidade de aplicativos e do sistema operacional. Um SDI efetua o registro de todas as tentativas de ataques, e com isso pode ser implementada uma correlação entre a freqüência e os tipos de tentativas de ataques. Essa correlação pode ser útil para prevenir novos ataques ao sistema que o SDI protege. Um SDI tem três funcionalidades principais, e a primeira delas é o monitoramento de informações. O SDI

pode monitorar diferentes origens de informações à procura de evidências de ataques em aplicações, máquina, ou uma rede de computadores. A segunda funcionalidade é efetuar a análise para detectar os eventos "suspeitos" nas fontes de informação. Essa análise pode ser através de regras ou detecção de anomalias. A terceira funcionalidade é a resposta ao incidente: quando os eventos evidenciam um ataque, então um conjunto de ações pode ser executado [14].

Nesse capítulo é apresentado um breve histórico dos SDIs. O processo de identificação de ataques é descrito, bem como uma classificação dos SDIs quanto aos meios utilizados para a detecção de ataques.

2.1 Breve Histórico de Sistemas de Detecção de Intrusão

O método mais popular e antigo de detecção de ataques é através de auditoria em arquivos de *log* de máquinas. A auditoria de *logs* foi o marco inicial para os Sistemas de Detecção de Intrusão atuais. O *National Computer Security Center*, no livro *A Guide to Understanding Audit in Trusted Systems*[1], define auditoria como "uma análise independente e exame de *logs* e atividades do sistema" [21]. No final da década de 70, administradores de redes faziam auditoria em *logs* de forma manual, o que consumia muito tempo além do elevado custo para armazenamento [46]. Em 1980, Anderson publicou um artigo que deu origem aos sistemas de detecção, intitulado *Computer Security Threat Monitoring and Surveillance* [11]. Neste artigo James Anderson introduz noções de auditoria de *logs* para detectar comportamento anômalo e noções sobre eventos específicos de usuários (comportamento) [16]. Entre 1984 e 1986, Dorothy Denning e Peter Neumann pesquisaram e desenvolveram o primeiro modelo de SDI de tempo-real. Em 1987, Denning publicou um artigo explicando a correlação entre detecção de anomalias e usos maliciosos dos sistemas. Seu protótipo de sistema de detecção é chamado de *Intrusion Detection Expert System* (IDES) [21, 16]. Em 1988, o projeto *Haystack* do Laboratório Lawrence Livemore da Universidade da Califórnia em Davis (UC Davis), desenvolveu outra versão de

[1] Da série de livros *Rainbow* do Departamento de Defesa Norte-Americano que descrevem e especificam vários aspectos de segurança computacional para o Governo Norte-Americano.

sistema de detecção para a Força Aérea Americana baseado em comparações de padrões pré-definidos. No final dos anos 80, os desenvolvedores do projeto *Haystack* criaram a empresa *Haystack Labs* e lançaram um SDI chamado *Stalker* [64], com novas tecnologias de análise de padrões para máquinas. No começo de 1990 a UC Davis introduziu a idéia de detecção de intrusão para redes, e Heberlein, da mesma universidade, foi o principal autor e desenvolvedor do *Network Security Monitor* (NSM), o primeiro SDI para redes [42]. Nos anos 90 a *Science Applications International Corporation* (SAIC) desenvolveu um SDI para máquina chamado de *Computer Misuse Detection System* (CMDS) [4]. Nessa mesma época o *Cryptologic Support Center* da Força Aérea Americana desenvolveu o *Automated Security Measurement System* (ASIM), um SDI para monitorar o tráfego da rede da Força Aérea Americana [28]. Posteriormente em 1994, desenvolvedores do ASIM formaram a empresa *Wheel Group* e lançaram, o *NetRanger*, um SDI para redes.

Desde então os SDIs começaram a ganhar popularidade, e em 1997 a *Internet Security System* (ISS) desenvolveu um SDI chamado de *RealSecure* [58]. No ano seguinte, a *Cisco Systems* comprou a *Wheel Group* para também entrar no mercado de SDIs. Desenvolvedores do CMDS e a *Haystack Labs* formaram a empresa chamada *Centrax Corporation* com uma versão de seu SDI. Desde então, várias empresas entraram no mercado de SDI e de segurança computacional [16].

2.2 Técnicas de Detecção de Intrusão

Os Sistemas de Detecção de Intrusão podem ser classificados em dois tipos, Sistemas de Detecção de Intrusão *Baseados em Regras*, e Sistemas de Detecção de Intrusão *Baseados em Detecção de Anomalias*. Ambos têm vantagens e desvantagens, e estas são discutidas no que segue.

2.2.1 Sistemas de Detecção de Intrusão Baseado em Regras

Nos Sistemas de Detecção de Intrusão *Baseados em Regras*, os dados, que podem ser o tráfego de uma rede ou arquivo de registro (*logs*) de uma máquina, são analisados à procura de padrões já conhecidos de um Banco de Regras (BR). O BR contém um conjunto de regras que definem padrões de ataques conhecidos, e esses padrões são formados por assinaturas de ataques, cada assinatura contendo uma descrição de um tipo de ataque. O sistema compara as assinaturas de ataques com os dados que estão em análise no momento. Essa comparação é executada através de cadeia de caracteres dos dados analisados com uma assinatura do BR. Se essa comparação for bem sucedida, então uma tentativa de ataque foi detectada, e um alerta é disparado ou uma ação é gerada em resposta ao incidente. Se essa comparação não for bem sucedida, então os dados analisados são comparados com a assinatura seguinte do BR, até que todas as assinaturas sejam comparadas aos dados analisados. Se nenhuma comparação for bem sucedida, então os dados analisados são processados pelo destino, sob a premissa de que estão "livres" de ataques. Essa técnica é similar à técnica utilizada em programas anti-vírus.

Sistemas de Detecção de Intrusão *Baseados em Regras* possuem algumas desvantagens. Quando há uma tentativa de ataque cujo padrão não é conhecido pelo BR, o ataque não é detectado. Portanto, para que o SDI seja efetivo, há a necessidade de uma constante atualização de regras para o BR. Se o BR for sempre atualizado existirão muitas regras a serem comparadas e com isso, o sistema perderá desempenho. Por outro lado, este sistema tem a vantagem de dificilmente indicar tentativas de ataques que não ocorreram.

Pode-se utilizar um Sistema Especialista [13] como mecanismo de busca por ataques conhecidos. A busca é dividida em duas fases: a fase de comparação de padrões de regras e a fase de ação. A fase de comparação consiste de auditoria de eventos, e a fase de ação é a resposta da fase anterior, e determina qual a é a ação que deve ser tomada quando as regras assemelham-se ao evento. O Sistema Especialista deve ser formulado por alguém que tenha conhecimento e experiência em segurança de computadores.

Um outro mecanismo é o Monitoramento de Teclas (*Keystroke Monitoring*) [65], que consiste em monitorar a atividade no teclado durante a sessão do usuário ou do atacante,

à procura de padrões de ataques conhecidos. A desvantagem deste mecanismo é que o usuário, ou atacante, pode criar atalhos para utilizar programas suspeitos com outros nomes. Só seria possível detectar esses atalhos se fossem analisadas a semântica dos comandos e as chamadas do sistema, para então analisar os programas que serão executados pelos usuários [65].

2.2.2 Sistemas de Detecção de Intrusão Baseado em Detecção de Anomalia

Sistemas de Detecção de Intrusão *Baseados em Detecção de Anomalia* detectam tentativas de intrusão através da observação de comportamento considerado anômalo. A distinção entre o "comportamento normal" e o "comportamento anômalo" requer conhecimento estatístico [35]. Um "comportamento normal" é expressado por perfis que representam o comportamento normal de usuários do sistema, de máquinas, ou de conexões de rede. Esses perfis são criados através de eventos coletados em um período considerado normal de utilização do sistema, e esse processo é chamado de treinamento, quando os perfis podem registrar o número de falhas no *logon* do sistema, a utilização de processamento e de memória, e uma lista de arquivos acessados [65]. Uma vez definidos os perfis normais, pode-se identificar violações destes [46].

Os perfis também podem ser definidos por um modelo que permite gerar variância de perfis já definidos. Tendo mais de um perfil, é possível fazer combinações de perfis e com isso o Sistema de Detecção de Intrusão *Baseado em Detecção de Anomalias* pode ser treinado gradualmente [65].

Uma outra maneira de definir perfis é através do modelo de previsão de eventos, que consiste em tentar prever eventos futuros com base em eventos ocorridos. Por exemplo, a regra de previsão de eventos: [E1 $\rightarrow$ E2 $\rightarrow$ (E3 = 80%, E4 = 15%, E5 = 5%)] define que caso já tenham ocorrido os eventos E1 e E2, e se E2 ocorreu depois do evento E1, então há 80% de chance que o evento E3 ocorra após o evento E2 ter ocorrido, ou 15% de chance que o evento E4 ocorra, ou há 5% de chance que E5 ocorra depois do evento E2. Com essa abordagem, se acontecer algum evento que não está previsto, este será considerado como

uma tentativa de ataque [65]. Esse modelo permite uma fácil detecção de atividades ou eventos anômalos, e pode-se associá-lo à técnicas de Inteligência Artificial, como Redes Neurais.

A desvantagem de SDIs *Baseados em Detecção de Anomalias* é o tempo de treinamento necessário para definir perfis, seja perfis de usuários, de conexões de rede ou de máquinas.

Sua principal vantagem está no reconhecimento de novas tentativas de ataques independentes de BR, o que pode ser utilizado para criar novas assinaturas de ataques para SDIs *Baseados em Regras.* Nesses sistemas de detecção de anomalias pode-se utilizar técnicas de Inteligência Artificial para reconhecer e aprender padrões [12]. Sua desvantagem é a quantidade de falsos positivos que, em determinadas situações, pode ser elevada, momentaneamente tornando seu uso pouco confiável.

Falso positivo é a detecção ou registro de um evento que não ocorreu, o falso positivo ocorre porque o sistema que gera esse evento conclui que uma determinada atividade foi executada. Na figura 2.1 é ilustrado um diagrama de veracidade de um evento ocorrido. Na esquerda superior da figura, é representado eventos normais que não são de ataques (verdadeiro negativo), já na esquerda inferior são ataques que não foram detectados (falso negativo). No lado direito superior do diagrama, representa ataques reais que foram detectado (verdadeiro positivo) e no lado direito inferior representa situação normal que é identificada como ataque (falso positivo).

Em Sistemas de Detecção de Intrusão, um falso positivo é um alerta relatado sobre uma suspeita infundada de ataque, como por exemplo um usuário de uma rede abre uma porta de comunicação de uma máquina para uma determinada ferramenta, o qual a porta aberta corresponde a uma utilizada em um ataque conhecido. O SDI gera então, um alerta infundado com informações sobre o tipo de ataque correspondente à porta aberta pelo usuário.

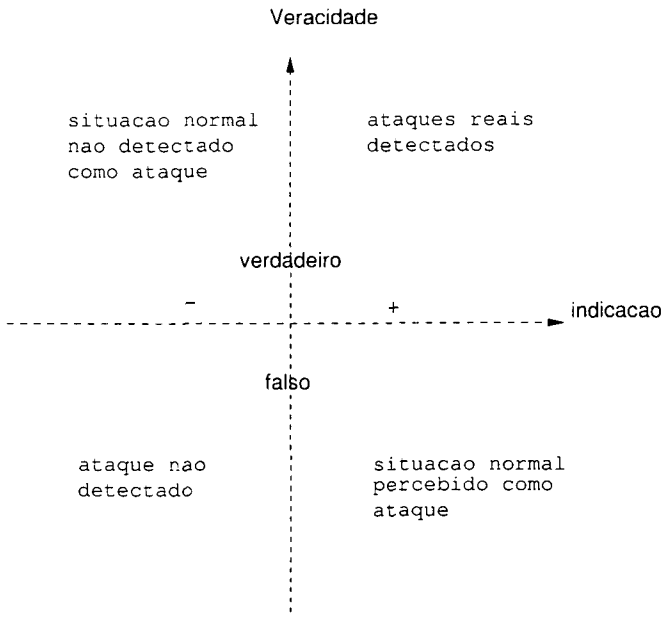

Figura 2.1: Diagrama de veracidade de alarmes.

Há também Sistemas de Detecção de Intrusão que são ao mesmo tempo, *Baseados em Regras* e *Baseados em Detecção de Anomalias* [12].

2.3 Classificação de Sistemas de Detecção de Intrusão

Os SDIs também podem ser classificados quanto a origem dos dados. Essa origem pode ser uma rede, no qual o SDI analisa o seu tráfego ou uma máquina, verificando os seus dados locais. Os Sistemas de Detecção de Intrusão que analisam tráfego de rede são chamados Sistema de Detecção de Intrusão para Rede (SDIR ou *Network-based Intrusion Detection System*) e os SDIs que analisam dados ou arquivos de registro locais em uma máquina são chamados de Sistema de Detecção de Intrusão para Máquinas (SDIM, ou *Host-based Intrusion Detection System*). Ambos têm vantagens e desvantagens, que são discutidas a seguir.

2.3.1 Sistemas de Detecção de Intrusão para Redes

Um SDIR examina o tráfego em um segmento da rede ou em toda rede. O tráfego é analisado através de uma interface de rede em modo promíscuo à procura de qualquer tipo de evidência de tentativa de ataque. Se for encontrada evidência, o sistema de detecção de intrusão cria um registro, em arquivos *log*, em forma de um alerta.

Uma técnica importante de detecção de intrusão *baseada em Regras* é através da análise de um protocolo específico para verificar se o tráfego do protocolo adere ao especificado no seu RFC (*Request For Comment*). Exemplos mais comuns desses protocolos são HTTP (*Hyper Text Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*) e Telnet. Qualquer violação do padrão descrito da RFC do protocolo é considerada como tentativa de ataque.

Sistemas de Detecção de Intrusão para redes modernas analisam também a camada 7 (camada de aplicação) do modelo OSI, e enquanto os SDIs antigos só analisavam a camada 3 (camada de rede) e a camada 4 (camada de transporte) do modelo OSI, que contém os protocolos IP (*Internet Protocol*), TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*).

Em um SDIR deve ser mantido um histórico das tentativas de ataques detectadas, para que seja feito um controle preventivo de possíveis ataques futuros. As principais características de SDIRs são:

- Análise de eventos da rede de forma independente da máquina ou conjunto de máquinas monitoradas;
- Ser executado em uma máquina dedicada à detecção de intrusão;
- Permite definir estratégias de análise em pontos críticos da rede, observando o tráfego destinado a um conjunto de máquinas;
- Ao contrário de um SDIM, não é possível que uma terceira parte altere os indícios de tentativas de ataques, já que a análise do tráfego é feita em tempo-real.

São duas as arquiteturas de SDIR: centralizada com um SDIR central, ou distribuída, que pode consistir em sensores, um balanceador de tráfego e um gerenciador desses sensores.

Os SDIRs possuem as seguintes desvantagens [14]. Quando há uma grande quantidade de tráfego a ser analisado, um SDIR centralizado pode falhar ao tentar detectar tentativas de ataques. Em uma rede segmentada por comutadores (*swicths*) não há grande eficiência do SDIR pois a maioria dos comutadores não possuem sistema de monitoramento de portas (este aspecto é tratado no capítulo 3) e assim o SDIR monitora uma quantidade limitada de tráfego, apenas de um segmento. O SDIR não pode analisar tráfego criptografado e muitos SDIRs não são capazes de concluir se uma tentativa de ataque foi ou não bem sucedida.

2.3.2 SDIRs Baseado em Sensores

Sensores, ou sensores SDIR, são Sistemas de Detecção de Intrusão distribuídos que analisam o tráfego da rede destinado especificamente a eles, ao contrário de um SDIR, que analisa todo o tráfego da rede. Esse tipo de SDIR pode ser empregado em redes de alta velocidade, em que um SDIR de arquitetura centralizada pode não ter a capacidade para analisar o tráfego da rede, sendo então necessária a distribuição do tráfego de modo que os sensores o analisem.

Sensores SDIR geralmente possuem duas partes: sensores propriamente ditos, responsáveis por analisar passivamente o tráfego destinado a eles, e um sistema de gerenciamento que é responsável pela resposta a um incidente e pelo controle dos sensores [8]. Nesse sistemas espera-se que reduzam o tempo e recursos computacionais envolvidos [71].

2.3.3 Sistemas de Detecção de Intrusão para Máquina

Sistemas de Detecção de Intrusão para Máquina são projetados para monitorar e detectar atividades inapropriadas em uma máquinas específica. Uma máquina alvo pode ser, por exemplo, um servidor crucial de uma rede. Como nos SDIRs, existem também dois tipos de SDIMs, *Baseados em Regras* e *Baseados em Detecção de Anomalias*. Os SDIMs

Baseados em Regras fornecem auditorias e suporte à evidências conhecidas, e é pressuposta a existência de uma BR. Todo SDIM pode desempenhar as seguintes tarefas: (1) detectar tentativas de conexões em portas não autorizadas, (2) monitorar e detectar acesso remoto indevido, (3) monitorar ações e horários de acesso do super-usuário, (4) monitorar arquivos de registro (*log*) de uma máquina e (5) monitorar o desempenho da máquina. Com essas cinco tarefas um SDIM pode gerar estatísticas que podem ser úteis para prevenir atividades maliciosas, tais como ataques do tipo negação de serviço local.

As duas principais desvantagens de um SDIM são: a possibilidade ser atacado e desabilitado como parte do ataque, e a sua execução pode interferir no processamento da máquina que ele está monitorando.

Há dois tipos de SDIM que serão descritos seguir, SDIM *Baseado em Pilha de Protocolo* e o SDIM *Baseado em Aplicativos.*

2.3.3.1 SDIM Baseado em Pilha de Protocolos

SDIM Baseado em Pilha de Protocolos é projetado para monitorar o tráfego entre a pilha TCP/IP e o sistema operacional ou aplicação, permitindo que o tráfego seja analisado na pilha TCP/IP. Desta forma o SDIM detecta tentativas de ataques antes que o tráfego chegue ao sistema operacional ou para a aplicação [49].

2.3.3.2 SDIM Baseado em Aplicativos

Um SDIM *Baseado em Aplicativos* é projetado para analisar eventos de um aplicativo específico, detectando atividades do aplicativo e a interação do usuário com o aplicativo, à procura de evidências de ações que possam comprometer a segurança do sistema computacional. Este tipo de SDIM pode monitorar a interação entre (a) o aplicativo e o usuário, (b) o aplicativo e o sistema operacional, e (c) o aplicativo e a quantidade de memória utilizada. SDI *Baseado em Aplicativos* fornece maior proteção a aplicativos por definir um perfil do uso comum da aplicação utilizando técnicas de *Detecção de Anomalia*, descrita na seção 2.2.2. Na fase de geração do perfil, o usuário tem que procurar utilizar todos os recursos do aplicativo de modo que o perfil seja treinado com a maior variedade

de interações entre o usuário e o aplicativo, evitando assim uma elevada taxa de falsos positivos [28]. Uma solução interessante para aumentar a segurança em uma Intranet, é utilizar SDI *Baseado em Aplicativos* combinado com SDIM, desta forma pode-se aumentar a proteção do sistema operacional e de aplicativos essenciais à Intranet [14].

2.3.4 Alertas Centralizados – Meta-SDI

Atualmente muitas organizações possuem vários Sistemas de Detecção de Intrusão, *firewalls*, sistemas de registro de *log*, cada qual com funcionalidades diferentes. Com o aumento na diversidade do tipo e no número de ataques, nenhum SDIR pode prover toda a segurança de uma rede [47]. Para se ter controle de todos os alertas reportados por diferentes SDIRs, SDIMs, diferentes *logs* de *firewalls*, há a necessidade de centralizar e padronizar alertas/*logs* em uma única base de dados.

O Meta-SDI gerencia os alertas/*logs* e extrai informações que revelam evidências de tentativas de ataques. Nele estão contidas as informações extraídas de alertas/*logs* em um padrão uniforme. Tentativas de ataques podem ser feitas a todos os pontos da rede, e o Meta-SDI visa correlacionar tentativas, por vezes isoladas, de ataques. Quanto mais informações forem enviadas ao Meta-SDI, mais eficiente ele se torna [47].

A figura 2.2 mostra um exemplo de arquitetura de Meta-SDI, consistindo de um SDIR, um sensor SDIR e um SDIM, todos eles centralizando seus alertas em uma só base de dados. Esses alertas são correlacionados pelo Terminal de Administração do Meta-SDI, que é responsável por extrair informações desses alertas. Os SDIs da figura 2.2 não precisam estar conectados diretamente ao Meta-SDI, mas devem ter algum mecanismo para que o Meta-SDI tenha acesso aos alertas reportados pelos SDIs. Uma possibilidade de padronização de alertas emitidos pelos SDIs é o formato descrito na seção 2.3.5, que proporciona a uniformidade dos alertas reportados ao Meta-SDI.

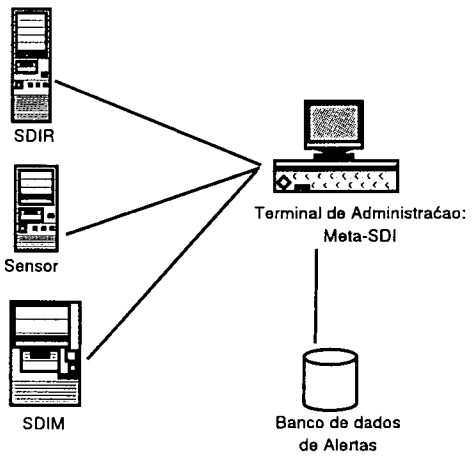

Figura 2.2: Exemplo de arquitetura com Meta-SDI.

2.3.5 IDEMF – *Intrusion Detection Exchange Message Format*

Organizações que empregam diferentes tipos de SDIs podem correlacionar alertas provindos desses SDIs para aumentar a segurança e prover respostas mais seguras e precisas aos incidentes. Há um formato padrão de troca de mensagens e alertas que permite que esses SDIs interajam.

O grupo IDWG (*Intrusion Detection Exchange Format Working Group*) da *Internet Engineering Task Force* [23, 37], desenvolveu uma especificação para formatos de alertas de SDIs. O intuito do IDWG é especificar um formato e procedimentos padrões para a troca de informações de interesse entre SDIs, ou entre SDIs e sistemas de resposta a incidentes.

O formato *Intrusion Detection Exchange Message Format* (IDEMF) é baseado em XML (*eXtensible Markup Language*) [52] e independe de protocolo de comunicação. O IDWG também propôs o IDXP (*Intrusion Detection eXchange Protocol*) [47], para servir de protocolo de transporte, mas não é necessária sua utilização para empregar o formato IDEMF.

2.3.6 Armadilhas

Armadilhas (*Deceptive Systems*) são sistemas projetados para detecção de intrusão em uma simulação real de um ambiente virtual. Essas Armadilhas foram primeiramente discutidas no *International Symposium on Recent Advances in Intrusion Detection* (RAID) de 1998 [3]. Armadilhas são sistemas que visam enganar o atacante de uma rede, de forma que, quando a política de segurança de uma rede é violada por um atacante, é realizado um redirecionamento do seu tráfego (sessão) pelo SDI para as Armadilhas [28]. Há vários tipos de Armadilhas, como *Padded Cell Systems*, *decoys*, *lures*, *fly-traps*, e Potes de Mel (*honeypots*) [40, 14, 63].

Atualmente, existem sistemas que implementam redes de Pote de Mel (*honeynets*) [10], e esses sistemas simulam uma rede com serviços falsos, que serve como uma Armadilha para o atacante [40]. Assim, o atacante pensa que está danificando uma rede "real" em um ambiente de produção, mas na verdade está em uma Armadilha, fornecendo com a tentativa de ataque, informações para auxiliar a resposta ao incidente e a identificação do atacante. A figura 2.3 mostra um exemplo de uma arquitetura de uma rede de Pote de Mel, com um *firewall*/SDI analisando o tráfego vindo da Internet. Quando o tráfego provindo da Internet é identificado como uma tentativa de ataque, o SDI redireciona esse tráfego para a rede simulada (rede de Pote de Mel), não afetando, assim, o ambiente de produção da rede. O tráfego que não compuser uma tentativa de ataque não é afetado. Uma Armadilha pode conter simulações de vários tipos de serviços, desde serviços básicos como FTP, ou HTTP, até aplicações de mais alto nível como servidores de banco de dados.

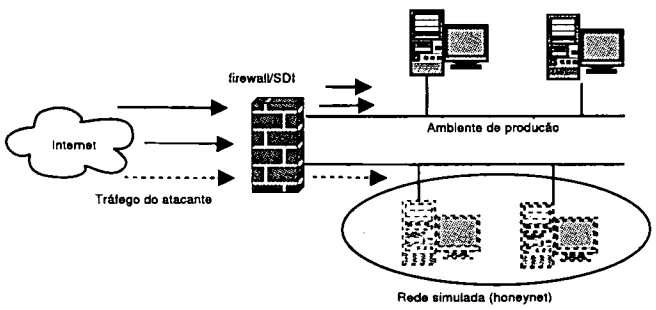

Figura 2.3: *Honeynet*: rede simulada

2.4 Conclusão

Sistema de Detecção de Intrusão são sistemas que analisam dados. Nesse capítulo foi introduzido as técnica que um SDI pode utilizar para detectar ataques, também foi apresentado tipos de SDI de acordo com a origem que analisa dados. No final do capítulo foi apresentado tópicos mais especializado em detecção de intrusão, como os sistemas Armadilhas.

Capítulo 3

Implementação de um Sistema de Detecção de Intrusão para Redes

Para implementar um SDIR, diversos aspectos devem ser considerados, destacando-se dentre aqueles o tipo de meio de conexão na qual se pretende implementar o SDIR, a quantidade de tráfego a ser analisada, políticas de segurança existentes, e a definição de novas políticas. Esta tarefa pode ser facilitada pelo entendimento das limitações do SDIR, das técnicas de detecção, bem como a capacidade de detecção do SDIR.

Esse capítulo está organizado da seguinte forma. Primeiramente são tecidas considerações sobre os meios e arquiteturas em que SDIRs podem ser implementados, e regras são descritas para a otimização de desempenho em um SDIR. No final do capítulo é descrita a distribuição de tráfego para uma arquitetura de detecção de intrusão distribuída e são apresentados experimentos com sobrecarga de SDIs.

3.1 Análise de Tráfego em um Meio Compartilhado de Conexão

Um meio compartilhado de conexão é uma rede no qual o meio físico é compartilhado entre todos os nodos da rede. Um concentrador (*hub*) é o dispositivo onde são conectados os nodos de uma rede com meio compartilhado. A implementação de um Sistema de Detecção de Intrusão para Redes em um meio compartilhado de conexão, através do uso de *hubs*, é efetuada simplesmente conectando o SDIR em uma das portas do concentrador. Com

isso, o SDIR pode analisar todo o tráfego da rede em que o concentrador está conectado.

3.2 Análise do Tráfego em uma Rede Segmentada

Uma rede segmentada é uma rede onde um comutador (*switch*) estabelece a ligação entre a porta de entrada (origem) e a porta de saída (destino) de forma dinâmica. O comutador é um dispositivo com características funcionais diferentes de um concentrador. No comutador a rede é segmentada em enlaces criados sob demanda. Com o uso de um comutador, a rede é segmentada por pares de portas, reduzindo o número de colisões[1], e aumentando a segurança e a privacidade.

Como um comutador segmenta a rede, a análise do tráfego por um SDIR pode ser comprometida, pois o SDIR somente pode verificar o tráfego da porta na qual está conectado. Para que SDIRs possam analisar todo o tráfego de uma rede segmentada é necessário utilizar um dos três mecanismos seguintes: concentrador, porta de espelhamento (*mirroring port*) e *test access port* (*tap*) [49], descritos a seguir.

3.2.1 Coletando Tráfego Através de um Concentrador

O concentrador pode auxiliar na coleta de todo o tráfego de uma rede segmentada para um SDIR analisar. Conecta-se o SDIR ao concentrador e pode-se conectar o concentrador entre dois comutadores, entre um servidor e um comutador, ou ainda entre um roteador e um comutador. Qualquer uma destas possibilidades permite que o tráfego que passa entre dois dispositivos seja copiado, chegando ao SDIR que está conectado no concentrador. Este tipo de coleta tem a desvantagem de apresentar uma alta taxa de colisões [49].

Na figura 3.1 é mostrada a utilização de um concentrador entre um roteador e um comutador. Conectado ao concentrador, está o SDIR que faz análise de todo o tráfego que passa entre o roteador e a rede que está conectada ao comutador.

[1] Quando um pacote é transmitido em em um meio compartilhado de conexão, o meio compartilhado entrega esse pacote para todos outros nodos. Caso dois nodos, que estam conectados ao meio, enviem pacotes ao mesmo tempo, os seus pacotes colidirão, resultando que em os nodos terão que retransmitir seu pacote.

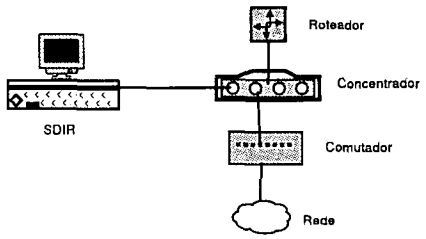

Figura 3.1: Uso de um concentrador para análise do tráfego em rede segmentada.

3.2.2 Porta de Espelhamento

Outra maneira de coletar todo o tráfego de uma rede segmentada é através do uso de uma *porta de espelhamento* nos comutadores, chamada também de porta *span* (*switch port analyzer*) [49]. A porta de espelhamento faz com que o comutador envie todo o tráfego da rede, ou de portas específicas, para uma porta de espelhamento configurada para receber este tráfego. Com isso, um SDIR pode ser ligado a esta porta e analisar todo o tráfego da rede segmentada. Esta maneira de coleta apresenta desvantagem semelhante àquela dos concentradores. Como a porta de espelhamento pode ter a mesma capacidade que as outras portas do comutador, o SDIR não é capaz de coletar todo o tráfego da rede, por sua limitação de capacidade. A porta de espelhamento também tem a desvantagem de não receber informações de VLAN (*Virtual* LAN) ou de pacotes de erros, além de somente receber fluxo de uma conexão *full-duplex* [49]. Estas desvantagens fazem com que o seu uso seja inadequado para a análise do tráfego por um SDIR.

3.2.3 Test Access Port

Test acess port (*tap*) é um dispositivo da camada 1 do modelo TCP/IP, que permite uma monitoração passiva do tráfego em um dispositivo de rede [49]. Para um SDIR coletar todo o tráfego de uma rede segmentada, pode ser utilizado um *tap* para obter passivamente todo o tráfego da rede. A implementação de dispositivo *tap* não requer modificação nas configurações da rede; um *tap* é um dispositivo separado de um comutador, mas também existem implementações de comutadores com *taps* embutidos. *Taps* têm a habilidade

de receber conexões *full-duplex*, de reduzir a perda de pacotes, além da possibilidade de coletar todo o tráfego transmitido em uma rede segmentada. O *tap* é utilizado também para aumentar a segurança de SDIRs, por ser um dispositivo passivo sem endereço IP, um *tap* pode receber diretamente todo o tráfego, e enviá-lo diretamente para a interface de rede do SDIR. Como não usam endereço IP, um *tap* previne ataques contra SDIRs pois esconde a existência de um SDIR na rede [33].

3.3 Melhoria de Desempenho de Sistemas de Detecção de Intrusão para Redes

Em redes de alta velocidade, desempenho é um fator essencial. Para que um SDIR possa garantir a confiabilidade do tráfego da rede analisada, são necessários procedimentos que visem otimizar a análise do tráfego por um SDIR.

Para a otimização do desempenho em SDIRs, deve-se levar em consideração: (1) o tráfego analisado, (2) a atividade das máquinas e (3) as tentativas de ataques que podem causar os maiores danos à rede. Também deve-se considerar a duração e a persistência da tentativa de ataque, pois como regra, quanto mais tempo a tentativa de ataque persistir, maior poderá ser o dano causado [21]. Há várias alternativas para aumentar o desempenho de um SDIR. Serão apresentadas algumas a seguir [27].

3.3.1 Políticas

Uma boa implementação de um SDIR depende da política de análise utilizada. Para otimizar a análise de tráfego, é comum o uso de políticas de análise, tais como não detectar tentativas de ataques a um serviço de FTP (*File Transfer Protocol*) em uma rede que não o disponibilize [26]. Detectar todas as tentativas de ataques ao sistema operacional Linux [56], em uma rede baseada no sistema operacional Windows [57], é um exemplo de uma política que pode ser considerada mal definida, pois há perda de tempo na análise pelo SDIR das tentativas de ataques ao sistema operacional Linux, quando essas tentativas não podem danificar a rede baseada no sistema operacional Windows.

3.3.2 Filtros

Filtros podem otimizar o desempenho de um SDIR porque evitam a análise de tráfego inofensivo, e o SDIR não perderá tempo de processamento com tráfego que não compromete a segurança da rede. Um cenário para filtragem de tráfego é, por exemplo, uma rede em que há uma máquina que emita grande quantidade de tráfego; este montante de tráfego não causa danos à rede, pois se trata de tráfego normal. Desta forma, cria-se um filtro para que o SDIR não analise o tráfego gerado por essa máquina, otimizando o desempenho do SDIR. Outro cenário para a filtragem de tráfego é uma rede com um conjunto de máquinas que se comunicam apenas por tráfego criptografado, através dos protocolos SSH (*Secure Shell*) ou SSL (*Secure Socket Layer*). Neste caso, não há necessidade do SDIR analisar este tráfego criptografado, pois o Sistema de Detecção de Intrusão não pode decodificá-lo.

3.3.3 Filtro para Interface de Rede

Geralmente, quando se implementa um SDIR, uma interface de rede adicional é utilizada para seu gerenciamento. Em uma rede segmentada por um comutador ou em meio compartilhado de conexão, pode-se ter vários pacotes irrelevantes que chegam à interface de gerenciamento do SDIR. Por exemplo, conforme o padrão 802.2 [67], o comutador pode enviar pacotes *multicast* e *broadcast* para todas as interfaces de rede do seu domínio [27]. Todas as interfaces de rede conectadas ao comutador devem examinar endereço MAC (*Medium Access Control*) destino para saber o que fazer com o mesmo (ignorar ou passar para a pilha TCP/IP). Isso poderia resultar em tráfego adicional irrelevante para o SDIR.

Pacotes *multicast* podem ser distribuídos erroneamente por um comutador para a interface de gerenciamento do SDIR caso a rede sofra uma sobrecarga de tráfego. Pacotes *multicast* têm conteúdo geralmente maior que pacotes *broadcast* ocasionando desta forma, perda de tempo na análise realizada pelo SDIR [27].

Pacotes do protocolo ARP (*Address Resolution Protocol*) do tipo *request* são menores que pacotes *multicast*, mas a maneira como são processados consome mais tempo. Isto poderia prejudicar o desempenho do SDIR em uma rede na qual haja grande quantidade

de ARP *requests*. Todos os pacotes ARP *request* são aceitos por todos os nodos da rede, cada um destes nodos traduz estes pacotes ARP para saber se o endereço IP do pacote ARP é o mesmo do nodo. Se o ARP *request* não for para o nodo específico, este é descartado, caso contrário, o nodo irá responder com ARP *reply*.

Dependendo do comutador empregado na rede, pode-se configurá-lo para bloquear ou limitar estes tipos de tráfego (*multicast*, *broadcast* e *unicast*), reduzindo assim o tráfego na interface de gerenciamento do SDIR.

3.3.4 Modificação de Temporizadores Padrão

O protocolo TCP têm temporizadores (*timeouts*) de retransmissão. Protocolos como UDP e TCP, ou protocolos mais específicos como HTTP (*Hypertext Transfer Protocol*), DNS (*Domain Name System*) e SMTP (*Simple Mail Transfer Protocol*), podem ter seus temporizadores codificados para prevenir a retransmissão TCP e, com isso, não há desperdício de tempo para a retransmissão. Podem ser atribuídos diferentes temporizadores para diversos protocolos. Por exemplo, no protocolo HTTP, o temporizador poderia ser reduzido, pois seu tráfego geralmente é alto, enquanto o protocolo Telnet poderia ter um temporizador maior, devido a seu baixo tráfego [27]. Fazendo uma análise da quantidade de retransmissão de pacotes TCP que ocorre em uma rede, torna-se mais apropriada a definição de temporizadores para diferentes protocolos, além de diminuir o tráfego para o SDIR. Também é diminuído a quantidade de falsos positivos que seriam gerados pelo SDIR, pois se a quantidade de retransmissões TCP for elevada o SDIR poderá classificá-la como uma tentativa de ataque do tipo DoS (*Denial of Service*) [17].

3.4 Sobrecarga em SDIR Centralizado

Muitos Sistemas de Detecção de Intrusão para Redes falham sob condições extremas ou quando estão sobrecarregados, possibilitando a não detecção de alguns ataques. Sob alto tráfego, o SDIR não consegue lidar com todo o tráfego passado por ele, ficando incapaz de efetuar uma análise precisa dos dados. Isto ocorre porque a carga de processamento do

SDIR é proporcional ao tráfego analisado. Como o SDIR analisa o tráfego passivamente. ele não comprometerá o tráfego, mas sim a sua própria confiabilidade.

A velocidade das redes cresce mais rápido que a capacidade dos processadores [48]. Desempenho é um dos pontos cruciais em um SDIR, portanto devem ser levados em consideração fatores [32], tais como:

- Sistema operacional e hardware utilizado no qual o SDIR está sendo executado;
- Tamanho do pacote: o padrão é 1500 bytes, pacotes menores causam sobrecarga, pois há um consumo maior de processamento na leitura do cabeçalho, dados, *checksum*, do pacote;
- Quantidade de tráfego criptografado, pois o SDIR não pode analisar dados criptografados;
- Escolha de políticas de segurança. Em um Sistema de Detecção de Intrusão *Baseado em Regras* seu desempenho também está associado a sua BR, quanto mais regras a serem analisadas, mais tempo de processamento será consumido.

Há vários pontos que determinam o desempenho de um SDIR. Experimentos reportados em [32] indicam que em redes de 10/100Mbps somente de 60 a 80% tráfego de entrada é analisado por um SDIR. Já em uma rede *Gigabit*, a taxa cai para 40-60% do tráfego.

Champion e Denz, em [20] apresentam uma comparação entre dois SDIRs, comparando as quantidades de ataques registrados por eles. Na realização dos testes, foram utilizados um SDIR comercial *Baseado em Regras* e um SDIR não comercial baseado em técnicas estatísticas (*Detecção de Anomalias*), desenvolvido pelo DARPA[2] e denominado *Ebayes*. Os ataques simulados foram do tipo negação de serviço, e para incitar os ataques foi usado um sistema que simula ataques por várias máquinas. Dessa forma construiu-se um ambiente que fornece uma simulação de situação real do ataque. A detecção dos ataques foi registrada utilizando o método que consiste em um aumento no número de máquinas atacantes por intervalo de tempo, e quanto maior o número de máquinas atacando os dois

[2] *Defense Advanced Research Projects Agency*

SDIRs, maior o intervalo entre os ataques. A consistência dos testes varia de acordo com a quantidade de ataques por faixa de tempo.

Na tabela 3.1 há uma cópia dos resultados relatados em [20]. Na primeira coluna indica o número de máquinas atacantes na simulação, a segunda o intervalo em segundos dos ataques. As duas últimas colunas, da tabela, demonstram a fração média detectada (média de seis experimentos) dos dois SDIRs, sem a presença de tráfego que não fosse do ataque. A tabela ?? demonstram que os dois SDIRs tendem a falhar quando há várias máquinas os atacando ao mesmo tempo. *Ebayes* consegue identificar com mais eficiência os ataques efetuados a partir de 40 máquinas simultânea. Acima deste número os dois SDIRs praticamente não conseguem identificar os ataques.

# Máquinas	Ataques/segundo	Comercial Fração média de detecção	Ebayes Fração média de detecção
10	2,50	0,200	0,700
10	3,33	0,050	0,833
10	5,00	0,033	0,517
10	6,00	0,000	0,367
15	3,75	0,175	0,422
15	4,95	0,100	0,700
15	7,50	0,025	0,567
15	9,00	0,012	0,144
30	7,50	0,033	0,300
30	15,00	0,021	0,478
30	18,00	0,011	0,000
40	10,00	0,050	0,325
40	13,32	0,000	0,708
40	20,00	0,005	0,000
40	24,00	0,011	0,000
60	15,00	0,033	0,000
60	20,00	0,025	0,000
60	30,00	0,008	0,000
60	36,00	0,000	0,000

Tabela 3.1: Experimentos de detecção de ataques por dois SDIs.

São descritos em [48], dois experimentos utilizando o *Snort*. Os experimentos foram realizados utilizando o tráfego produzido por *Mit Lincoln Labs* [55]. O *log* do tráfego foi injetado em uma rede *Gigabit* com a ferramenta *tcpreplay* [7]. O primeiro experimento foi executado utilizando 18 regras no BR, com tráfego que varia de 20Mbps a 200Mbps. O número de tentativas de ataques foi constante. Quando o tráfego atingiu a faixa de

130-150 Mbps houve decréscimo no número de alertas emitidos pelo Snort. No segundo experimento o fator variante foi o número de regras do BR, a um tráfego de 100Mbps, também com um número fixo de tentativas de ataques. Constatou-se que ao aumentar o número de regras para a detecção de intrusão houve queda no número de alertas reportados. Este segundo experimento demonstra que quanto mais assinaturas no BR, menor será o desempenho de um SDIR que o utiliza.

Um conjunto de experimentos foi realizado por nós para avaliar a eficiência de SDIRs para diferentes quantidades de tráfego, em um SDIR centralizado.

Os experimentos descritos abaixo empregam o SDIR de domínio público Snort [50] e fazem parte da contribuição deste trabalho. As tentativas de ataque foram realizadas em intervalos aleatórios, mas com o tráfego constante. A arquitetura utilizada é ilustrada na figura 3.2, a qual mostra o uso de um comutador como meio de interligação entre o "Atacante", que é responsável pelo ataque e pela geração do tráfego, e o servidor *proxy*, que é responsável pelo redirecionamento de requisições HTTP e pelo Sistema de Detecção de Intrusão. Conectado ao *proxy* está o servidor HTTP, que aceita as requisições HTTP do atacante. O ataque consiste em tentativa de acesso a conteúdo proibido do servidor HTTP. Para executar os testes foram utilizados dois servidores conectados em uma rede de 100Mbps. Como servidor *proxy* foi utilizado um servidor Intel Pentium III de 800Mhz com 256 MB RAM, e como servidor HTTP um computador Intel Pentium IV de 2Ghz com 256 MB RAM.

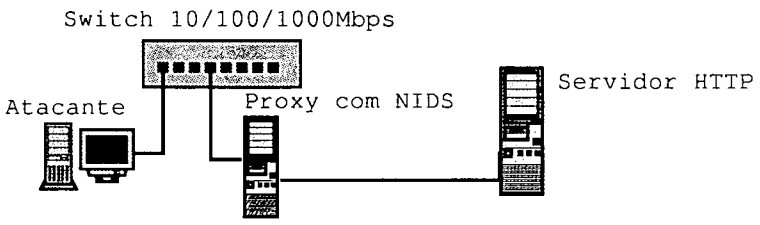

Figura 3.2: Arquitetura utilizada para a realização dos experimentos.

! O tráfego utilizado foi induzido pela ferramenta hping2 [44]. O Snort foi configurado com apenas uma regra. Visando avaliar o impacto da freqüência dos ataques, não houve requisições de abertura de conexão TCP falsa de forma a isolar os problemas causados pela velocidade da rede. Foram utilizadas cinco taxas diferentes de velocidade de rede. A figura 3.3 ilustra os resultados obtidos. Observou-se que nas taxas entre 28Mbps a 50Mbps o Snort funcionou adequadamente, e conseguiu reportar as tentativas de ataques com quase 80% de eficiência. Nos testes a 80Mbps o Snort não funcionou adequadamente, pois não reportou mais do que 30% das tentativas de ataque. Na faixa de 90-100Mbps o Snort se mostrou praticamente inoperante, pois não detectou nenhum ataque.

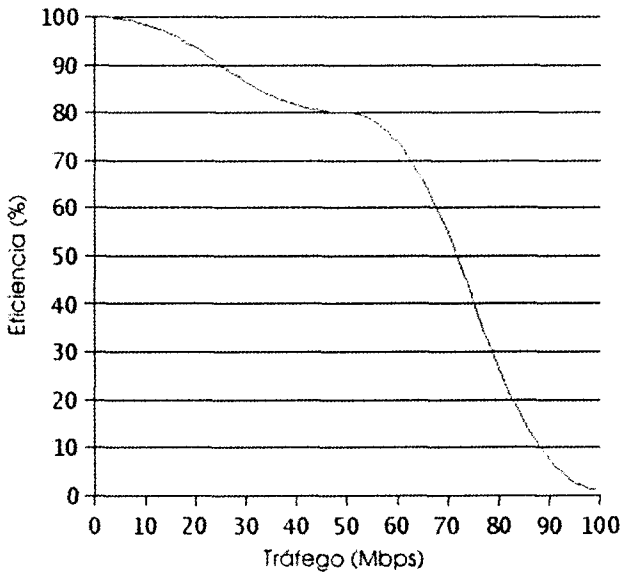

Figura 3.3: Demonstração da ineficiência do Snort.

Os experimentos descritos acima demonstram que uma análise de alto tráfego por um SDIR centralizado requer um poder de processamento que acompanhe esse tráfego. Isto implica que é necessária a distribuição desse tráfego para que seja feita a análise deste, evitando desta forma a sobrecarga em somente um SDIR centralizado.

3.5 Balanceamento de Carga

Com o aumento da velocidade das redes, com o aumento de utilização de sistemas computacionais paralelos um fator importante em sistemas distribuídos é a maximização de processamento por balanceamento de carga [60].

Balanceamento de carga é um mecanismo que tem como objetivo maximizar o desempenho de uma certa aplicação, através da divisão da carga de forma igualitária, para que os nodos efetuem a fração da carga.

O balanceamento de carga pode aumentar o desempenho de uma aplicação, distribuindo o processamento desta em (a) diversos nodos que fazem parte do sistema de balanceamento de carga, ou (b) para um nodo com poder de processamento maior, ou (c) para um nodo que tenha recursos que outros nodos não disponham [19, 53]. A melhora no desempenho de uma aplicação que o balanceamento de carga possa proporcionar, é provida por julgamentos (escalonamentos) da distribuição ou redistribuição de cargas. Esses julgamentos são executados por algoritmos de balanceamento de carga que desempenham a decisão da distribuição de carga para nodos de uma rede. Os algoritmos para balanceamento de carga podem ser classificados em dinâmicos e estáticos. Um algoritmo dinâmico para balanceamento de carga utiliza informações dos nodos para distribuição de tarefas; como por exemplo, informações quanto à carga de processamento de um nodo. Um algoritmo estático para balanceamento de carga não utiliza tais informações. As vantagens do balanceamento de carga são maior desempenho, disponibilidade e escalabilidade [53, 70].

3.5.1 Arquitetura de uma Rede com Balanceamento de Carga

A arquitetura de um sistema de balanceamento de carga para rede consiste em três elementos básicos:

- Usuários - são clientes que fazem requisições ao balanceador de carga;
- Balanceador de carga - é um servidor que executa o algoritmo de balanceamento de carga das requisições provindas dos usuários para os servidores reais que fazem

parte do sistema de balanceamento de carga;

- Servidores reais - são servidores que atendem às requisições dos usuários, distribuídas pelo balanceador de carga.

Esta arquitetura tem a vantagem de prover redundância: se um dos servidores reais falhar, não haverá comprometimento total da aplicação; na falha de um servidor real, o serviço provido terá alguma diminuição no seu desempenho. Caso a aplicação exija alta capacidade de processamento, ou recursos somente disponíveis no servidor real falho, a aplicação ficará comprometida. Na figura 3.4 é mostrada uma arquitetura que utiliza balanceamento de carga. O usuário faz uma requisição ao balanceador de carga, e este irá distribuir a carga imposta pela requisição entre seus servidores reais. O círculo pontilhado ao redor de *Balanceador de Carga*, *Servidor 1*, *Servidor 2* e *Servidor 3*, que representam os servidores reais, inclui todo o sistema de balanceamento de carga e que pode ser chamado de Servidor Virtual. Todo o conjunto de balanceador de carga e servidores que atendem às requisições será visto por seus usuários como sendo um servidor virtual.

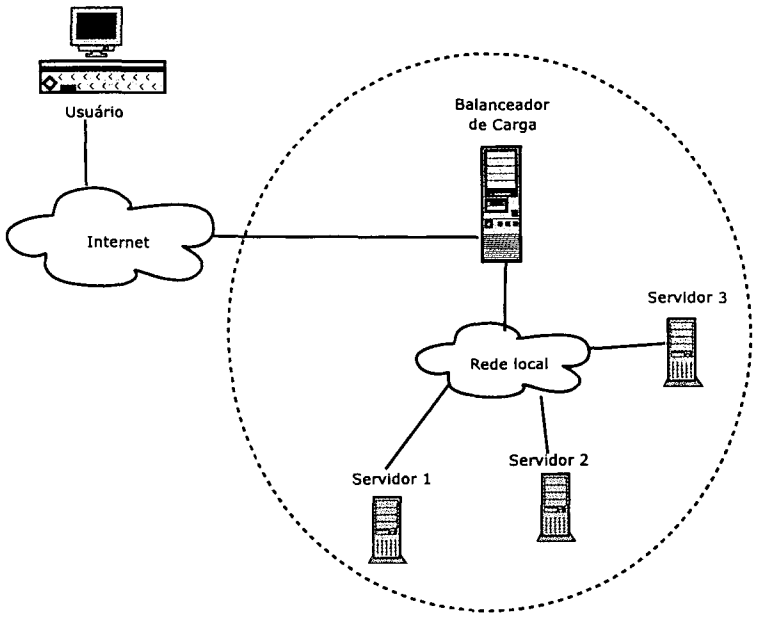

Figura 3.4: Arquitetura de uma rede com balanceamento de carga.

Em uma arquitetura com um sistema com o balanceamento de carga existe também a vantagem de se poder adicionar novos servidores reais de forma transparente aos usuários [25]. Existem algoritmos de balanceamento de carga os quais utilizam "pesos" em cada um dos servidores reais, diferenciando servidores reais com maior capacidade de processamento, ou com recursos diferentes, de outros servidores reais. Para exemplificar este algoritmo utilizando pesos, na figura 3.4, supondo que o *Servidor 1*, tenha o dobro da capacidade de processamento dos outros dois servidores reais, existe a possibilidade de atribuir ao *Servidor 1* um peso que indique que ele pode suportar maior carga ao atender às requisições provindas dos usuários. Uma aplicação do algoritmo de balanceamento de cargas é o balanceamento de tráfego, que é discutida na próxima seção.

3.5.1.1 Balanceamento de Tráfego

O balanceamento de tráfego é uma das aplicações do balanceamento de carga. Pode-se obter o balanceamento de tráfego através de comutadores, ou através de roteadores, que implementem algoritmos para balanceamento de tráfego. Existem comutadores/roteadores que trabalham até a camada de aplicação do modelo TCP/IP. Com isso, podem segmentar ou rotear o tráfego de acordo com o protocolo da aplicação. Estes dispositivos, permitem distribuir o tráfego para enviá-lo diretamente a sensores SDIR. Com essa facilidade, pode-se distribuir o tráfego de forma que cada sensor receba a mesma quantidade de tráfego independente dos protocolos/serviços, ou divide-se o tráfego por serviço ou por protocolo. A figura 3.5 mostra uma rede com três serviços, e o tráfego é dividido na mesma proporção por um comutador para sensores SDIR. Nesta divisão, todos os sensores executam as mesmas políticas (regras e filtro) e, assim, são capazes de detectar as mesmas tentativas de ataques.

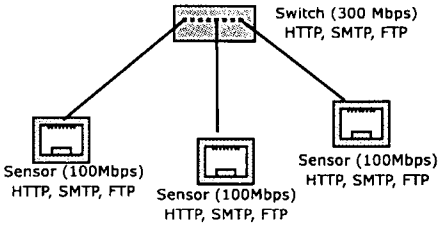

Figura 3.5: Distribuição igual de tráfego para sensores SDIR.

A figura 3.6 mostra a segunda abordagem, na qual o tráfego é dividido de acordo com o protocolo. Na figura é ilustrado o tráfego do protocolo HTTP, redirecionado para o *Sensor 1*, o tráfego dos protocolos FTP e SMTP são redirecionados para o *Sensor 2* e o tráfego do protocolo RPC (*Remote Procedure Call*) é analisado pelo *Sensor 3*. Assim, cada sensor tem políticas e regras diferentes de acordo com o protocolo ou serviço analisado, e cada um detectará os tipos de ataques do tráfego do protocolo.

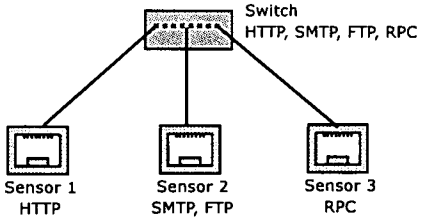

Figura 3.6: Distribuição de tráfego por protocolo para sensores SDIR.

A divisão de tráfego por protocolo pode-se tornar ineficaz. Quando há uma grande quantidade de tráfego de um protocolo específico e de outros um quantidade pequena de tráfego, pode haver uma sobrecarregar em um SDIR. Na divisão igual do tráfego se acontecer uma sobrecarga, acontecerá em todos SDIRs, que fazem parte dessa análise distribuída, ficando necessário a inclusão de um ou mais SDIRs para tal análise.

3.5.1.2 Experimentos com Balanceamento de Tráfego para SDIR Baseado em Sensores

Experimentos descritos em [9] demonstram a eficiência do balanceamento de tráfego na detecção de tentativas de ataques por sensores SDIR. Os experimentos mostram que à medida que o tráfego aumenta, é preciso que haja um número maior de sensores SDIR para o analisar. Nos experimentos foram utilizados de 1 a 9 sensores. Cada número de sensores foi testado com tráfego composto por pacotes de diferentes tamanhos, 50% do total de tráfego foi produzido com pacotes de tamanho de 1514 bytes, 40% com pacotes de 512 bytes e 10% com tamanho de 64 bytes. O tráfego total varia de 100 Mbps até próximo de 1000 Mbps. Tais experimentos foram feitos com um SDIR Baseado em Sensores chamado *RealSecure Network Sensor* da *ISS* [58].

O resultado desse experimento mostrou a relação da quantidade de tráfego distribuído por eficiência na detecção de ataques, foi mostrado que, o valor da eficiência da detecção é proporcional ao número de sensores analisando o tráfego distribuído. Para uma detecção de ataques simulados a uma velocidade 1000 Mbps foi necessário distribuir esse tráfego a 9 sensores para conseguir uma taxa de 84% da detecção dos ataques simulados.

3.6 Conclusão

Nesse capítulo foram vistos algumas considerações sobre a implementação de um SDIR, como o meio o qual o SDIR analisa o tráfego, pontos relevantes sobre a melhoria no desempenho de um SDIR para aumentar sua chance de detecção e confiabilidade. Foi também demonstrando experimentos de sobrecarga de um SDIR centralizado analisando o tráfego de uma rede de alta velocidade. No final desse capítulo foi apresentado soluções para analise do montante de tráfego em uma rede de alta velocidade.

Capítulo 4

Nariz – Um Sistema de Correlacionamento Distribuído de Alertas

A detecção de intrusão em redes de alta velocidade tem se mostrado eficiente em ambientes de detecção distribuída Baseados em Sensores [48, 9]. Nestes sistemas, cada sensor é responsável por analisar passivamente uma fração do tráfego e por emitir um alerta caso um sensor detecte uma tentativa de ataque.

Em redes de alta velocidade sensores SDIR tendem a emitir uma quantidade de alertas maior que em redes locais, pois o tráfego é maior, e portanto, estas redes podem sofrer mais tentativas de ataques por unidade de tempo. A vazão elevada pode produzir uma quantidade maior de alertas provindas de repetições e de uma taxa mais elevada de falsos positivos [22].

Neste capítulo é apresentado um sistema de correlacionamento de alertas de SDIRs em uma rede de alta velocidade TCP/IP, utilizando dois modos de correlacionamento, o *correlacionamento de eventos* e o *correlacionamento de alertas*, que são introduzidos a seguir. No apêndice A está descrito o algoritmo utilizado para o correlacionamento distribuído.

4.1 Correlacionamento de Eventos

O correlacionamento de eventos é uma técnica utilizada na gerência de redes para monitorar o comportamento de uma rede ou de componentes da rede, objetivando identificar padrões de eventos que possam significar ataques, intrusões ou falha em algum sistema [45, 41]. Componentes de redes geralmente emitem dados que informam o seu estado atual e tais informações de estado são armazenadas em arquivos de registro (*logs*). Esses componentes e os dados produzidos por eles podem ser, por exemplo:

- Servidores *web*, que podem emitir dados a cada requisição a uma página, como também podem emitir dados a cada erro de requisição;
- Aplicações na rede, que podem gerar *logs* indicando erros, avisos ou falhas na própria aplicação;
- *Firewalls*, que podem emitir dados em forma de arquivo registro, indicando uma tentativa de acesso não autorizado;
- Roteadores podem emitir informações em *logs* quando chega à rede uma determinada quantidade de tráfego;
- Comutadores e concentradores, que podem informar algum erro de configuração.

Os dados emitidos por esses componentes são chamados de *eventos* e cada evento informa uma mudança de estado de um determinado componente em um dado instante. Em uma rede existem outros componentes que também emitem eventos, como por exemplo, servidores de banco de dados, SDIs ou até mesmo o próprio sistema operacional de algum servidor que faça parte da rede.

Um administrador de rede humano não consegue acompanhar uma grande quantidade de eventos, devido ao processo manual que o humano teria que realizar e devido às limitações físicas. O objetivo principal do correlacionamento de eventos é reduzir a quantidade de eventos produzidos pela rede e condensá-los para que o administrador da rede possa analisar a informação contida nos eventos [68]. A diminuição na quantidade de

eventos pode ser atribuída a quatro técnicas principais de correlacionamento de eventos, que são a compressão, a priorização, a generalização e o correlacionamento baseado em tempo [45]. Na técnica de *correlacionamento por compressão* são descartadas as várias ocorrências do mesmo evento, eliminando assim as informações redundantes dos eventos "duplicados" [62]. A técnica de *correlacionamento por priorização* depende da atribuição de prioridade a eventos considerados mais importantes, e que devem ser examinados primeiramente. Na técnica de *correlacionamento por generalização*, o correlacionamento é realizado através da análise de eventos do mesmo tipo, mas de diferentes componentes da rede, consolidando esses eventos em uma mensagem constituída de informações relevantes aos diferentes componentes. A técnica de *correlacionamento baseado em tempo* depende da hora em que diferentes eventos ocorrem. Essa técnica é utilizada para obter informações de potenciais eventos (causa) que foram associados a uma falha, ataque ou intrusão, em um dado momento.

Para entender melhor as quatro técnicas de correlacionamento, aplicaremos cada uma delas nos eventos gerados por um servidor de hospedagem de páginas. Os eventos mostrados na Figura 4.1 apresentam exemplos de registros de eventos gerados nesse servidor por serviços de SSH (*Secure Shell* - acesso remoto criptografado), FTP e *Web* (servidor de páginas). Os eventos são apresentados na seguinte organização: número identificador do evento, serviço e descrição do evento gerado, hora e data do acontecimento do evento. Os eventos de número *11*, *15*, *18* e *19*, representam um tipo de ataque ao serviço de SSH, descrito em [34].

Utilizando a técnica de *correlacionamento por compressão*, por exemplo, seriam descartadas as várias ocorrências do evento "WEB: Falha de acesso - usuário: thiago" – eventos *12*, *13*, e *14*, e estes três seriam comprimidos em um só evento. Aplicando a técnica de *correlacionamento por priorização* seria atribuída ao evento de número *20*, por exemplo, uma prioridade mais alta que a dos demais, assim este seria analisado primeiramente. Usando a *correlação por generalização* poderiam ser agrupados todos os eventos de "Falha de acesso", eventos *12*, *13*, *14*, *16* e *17*, em somente um evento contendo informações sobre os tipos de eventos e as diferenças entre eles. Utilizando a técnica de *correlacionamento*

baseado em tempo pode-se observar quais foram os eventos que ocorreram entre as 17 e 18 horas, que são todos os eventos listados na figura 4.1.

```
...
11 - SSH: Usuário inválido: 'AAAA.%24$08X'@localhost -- 17:17-06/05/2004
12 - WEB: Falha de acesso - usuário: thiago -- 17:37-06/05/2004
13 - WEB: Falha de acesso - usuário: thiago -- 17:37-06/05/2004
14 - WEB: Falha de acesso - usuário: thiago -- 17:37-06/05/2004
15 - SSH: Usuário inválido: 'AAAA.%24$08X'@localhost -- 17:39-06/05/2004
16 - FTP: Falha de acesso - usuário: roberto -- 17:38-06/05/2004
17 - WEB: Falha de acesso - usuário: roberto -- 17:38-06/05/2004
18 - SSH: Usuário inválido: 'AAAA.%24$08X'@localhost -- 18:27-06/05/2004
19 - SSH: Usuário inválido: 'AAAA.%24$08X'@localhost -- 18:27-06/05/2004
20 - SSH: CRÍTICO - Não responde - CRÍTICO -- 18:27-06/05/2004
...
```

Figura 4.1: Exemplo de registro de eventos gerados pelo um servidor de hospedagem de páginas.

O correlacionamento de eventos é utilizado também na área de tolerância a falhas, gerenciamento de segurança e detecção de intrusão. Na detecção de intrusão a correlação é realizada através de alertas emitidos por SDIRs [26].

4.2 Correlacionamento de Alertas Emitidos por Sensores SDIRs

Uma estratégia para diminuir a quantidade de alertas emitidos por sensores SDIR é a utilização de um sistema que correlacione alertas provindos de diferentes sensores SDIR com intuito de reduzir o número de alertas e de falsos positivos, permitindo ainda que se obtenha mais informações sobre uma suspeita de ataque [43, 15].

Toda vez que um sensor emite um alerta, o sistema de correlacionamento obtêm esse alerta e o grava em uma base de dados. Antes de o sistema gravar o alerta é executada uma rotina de correlacionamento que implementa umas das técnicas mencionadas na seção 4.1.

Como exemplo de uma regra de correlacionamento de alertas, suponha que o atacante

tentou enviar um vírus para todos os servidores de uma rede `x.y.z`. Como o tráfego é particionado de modo que cada sensor analise uma porção do tráfego (seção 3.5.1.1), cada sensor identifica a tentativa de ataque pelo mesmo atacante e cada um dos sensores emite um alerta. O sistema de correlacionamento de alertas, após inserir em sua base um alerta emitido por um dos sensores, evitará a duplicidade de alertas com a detecção desta mesma tentativa de ataque, reduzindo-se assim o número de alertas que o administrador da rede terá de analisar [22].

Sistemas de correlacionamento de alertas também podem utilizar técnicas de análise estatística utilizando características como endereço IP destino e origem, ou porta TCP. Uma outra técnica emprega sistemas especialistas que podem aprender através de alertas que sempre ocorrem e que são falsos positivos [22, 66].

A quantidade de alertas emitidos por SDIRs em uma rede de alta velocidade pode ser elevada. Um sistema de correlacionamento de alertas pode ter dificuldades para relacionar um número elevado de alertas em tempo real da mesma forma que um SDIR centralizado perde desempenho ao analisar todo o tráfego de uma rede de alta velocidade, como discutido na seção 3.4. Para o correlacionamento dos alertas emitidos por sensores em uma rede de alta velocidade, em tempo real, é necessário que o sistema de correlacionamento de alertas trabalhe de forma distribuída. Esse trabalho de forma distribuída é realizado por dois processos de correlacionamento, o *correlacionamento local*, e o *correlacionamento distribuído* de alertas.

A correlação de alertas local já é utilizada em ambientes comerciais, como o correlacionador chamado *RealSecure SiteProtector Fusion Module* da ISS [2, 29]. Existem vários estudos sobre este tipo de correlacionamento, como por exemplo, o sistema de correlacionamento de alertas por probabilidade descrito por Alfonso Valdes *et al.* em [69]. A correlação distribuída de alertas é uma técnica nova e é o objeto deste trabalho. Esta técnica é descrita na próxima seção.

4.3 Arquitetura de sensores SDIRs Baseados no Nariz

A arquitetura de correlacionamento distribuído de alertas é composta por três componentes principais, que são o espalhador, os sensores e os correlacionadores de alertas. O *espalhador* é o componente responsável pelo particionamento do tráfego, e o roteamento deste para os sensores. Os *sensores* são sistemas de detecção de intrusão para redes que analisam o tráfego que é direcionado a eles pelo espalhador. Os *correlacionadores* são sistemas que correlacionam alertas emitidos pelos sensores. Os alertas são correlacionados em duas fases, na primeira, é executado o correlacionamento local, e na segunda fase é realizado o correlacionamento distribuído através da troca de mensagens entre os correlacionadores.

A figura 4.2 mostra a utilização de três sensores analisando o tráfego, que é distribuído pelo espalhador. Cada um dos três sensores contém um Nariz para correlacionar alertas do próprio sensor (correlacionamento local) e correlacionar as mensagens emitidas pelos outros Narizes (correlacionamento distribuído). Os Narizes se comunicam através do canal de comunicação, que é representado pela linha vertical. e quando necessário, enviam mensagens ao administrador da rede, representado no canto superior direito da figura. A figura também mostra a ligação entre os sensores/Narizes.

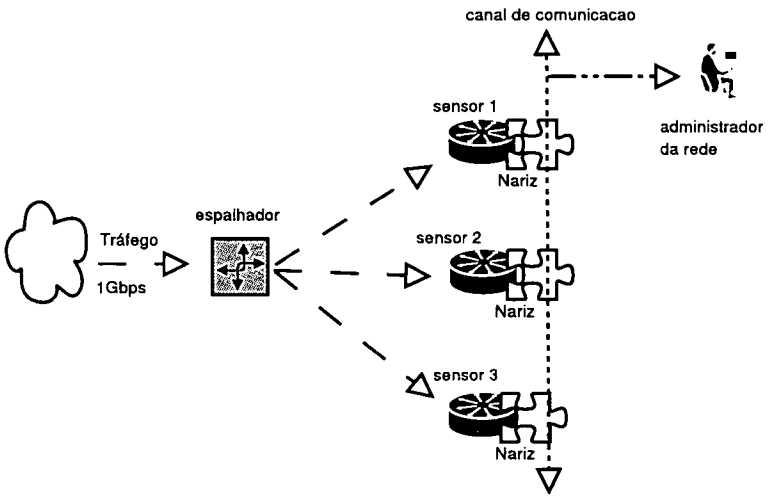

Figura 4.2: Arquitetura de sensores com o sistema Nariz.

4.3.1 Espalhador

Para permitir que SDIRs analisem o tráfego de redes de alta velocidade, o tráfego é dividido em "porções", e cada porção é analisada por um sensor SDIR ou por um conjunto de sensores. A divisão do tráfego é implementada no espalhador, por meio de distribuição de tráfego circular (*Round Robin*). Utiliza-se o espalhador de tráfego de redes de alta velocidade para sensores SDIRs descrito em [48]. Esse espalhador garante que o particionamento do tráfego mantêm a detecção de todos os tipos de ataques especificados. Cada porção de tráfego corresponde a uma sessão, e com isso o tráfego distribuído é baseado em conexões [9].

Modelo Lógico O modelo lógico do espalhador é constituído de um tap, um espalhador e distribuidores de tráfego. O tap é responsável por obter passivamente o tráfego de uma rede de alta velocidade e enviá-lo para o espalhador. A função do espalhador é dividir esse tráfego em seqüências de quadros e cada uma dessas seqüências é transmitida para um distribuidor diferente. Os distribuidores possuem a função de rotear os quadros que eles recebem para os sensores ligados a um ou mais canais de saída através de um comutador. Os canais de saída são associados a um conjunto de sensores que analisam os quadros transmitidos. O espalhador e os distribuidores são implementados por processos em um computador.

A figura 4.3 contém um diagrama do modelo lógico do espalhador. Na esquerda da figura é mostrado o tráfego da Internet, e os quadros são representados por Qs. O tap obtém o tráfego para o *espalhador*, que por sua vez particiona o fluxo de quadros circularmente em seqüências. Os *distribuidores* de tráfego, representados por *d0* a *d5*, recebem essas subseqüências de quadros (Qj) e roteiam os quadros para um conjunto de canais de saída (QCi). O quadrado no centro da figura contém o *espalhador* e os *distribuidores*, que são implementados como processos sendo executados em uma máquina. As saídas dos *distribuidores* (seqüência de quadros) são enviadas aos sensores ligados às portas do comutador, uma para cada canal de saída. Um ou mais sensores são ligados a cada canal de saída e estes são configurados para detectar certas classes de ataques.

Com a quebra do fluxo de quadros em seqüências, o tráfego a ser analisado por um sensor é apenas uma fração do fluxo original. Esta redução do tráfego possibilita o uso de um algoritmo mais sofisticado nos sensores, ou o emprego de máquinas de menor capacidade (e custo) nos sensores.

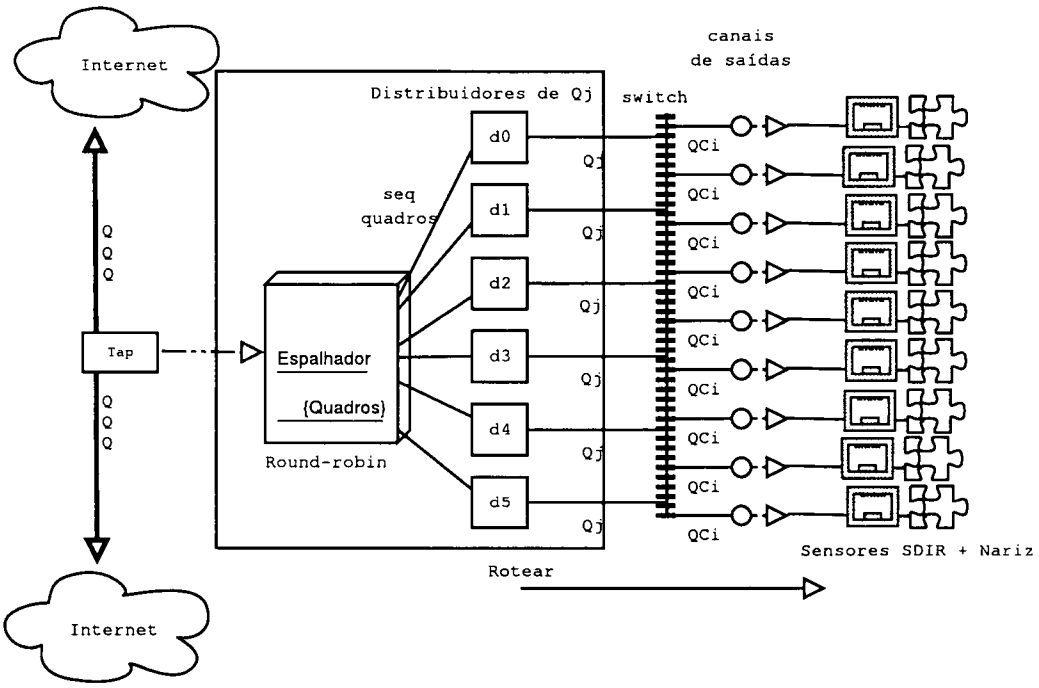

Figura 4.3: Modelo lógico do espalhador.

4.3.2 Sensores

Os sensores empregam o SDIR de domínio público chamado Snort [59, 50], que é baseado na biblioteca *pcap* [6]. O Snort pode emitir alertas de tentativas de ataques em tempo real e em diversos formatos. Seu mecanismo de detecção utiliza uma arquitetura modular, possibilitando aos usuários utilizarem suas próprias extensões de pré-processadores de detecção, e regras de detecção [61]. As regras utilizadas pelo Snort permitem gerar alertas de tentativas de ataques, ou copiar os pacotes suspeitos (*log*), por exemplo. A arquitetura de correlacionamento distribuído utiliza sensores autônomos espalhados para a detecção de intrusão, e cada sensor reporta seus alertas ao Nariz local.

4.3.3 Nariz

O sistema de correlacionamento de alertas Nariz é constituído de três componentes: (1) correlacionamento de alertas local, que é executado no sensor SDIR em que o Nariz está implementado, (2) correlacionamento distribuído de alertas, cuja execução é baseada em troca de mensagens pelos Narizes e (3) base de alertas, onde são armazenados os alertas recebidos pelo Nariz local. O sistema possui regras simples de correlação de alertas emitidos por sensores. Os alertas não correlacionados são inseridos na base como alertas novos.

Conforme ilustra a figura 4.4, os correlacionadores obtém os alertas dos sensores SDIR, correlacionam localmente em sua base de alertas e se comunicam para executar o correlacionamento distribuído através de troca de mensagens. A troca de mensagens é realizada através do canal de comunicação, e são enviadas mensagens do emissor para todos os Narizes que fazem parte do sistema de correlacionamento, de modo que todos os Narizes recebam as mesmas mensagens. O correlacionamento dos alertas é realizado através de variáveis de correlacionamento e aplicação de regras de correlacionamento, descritas a seguir.

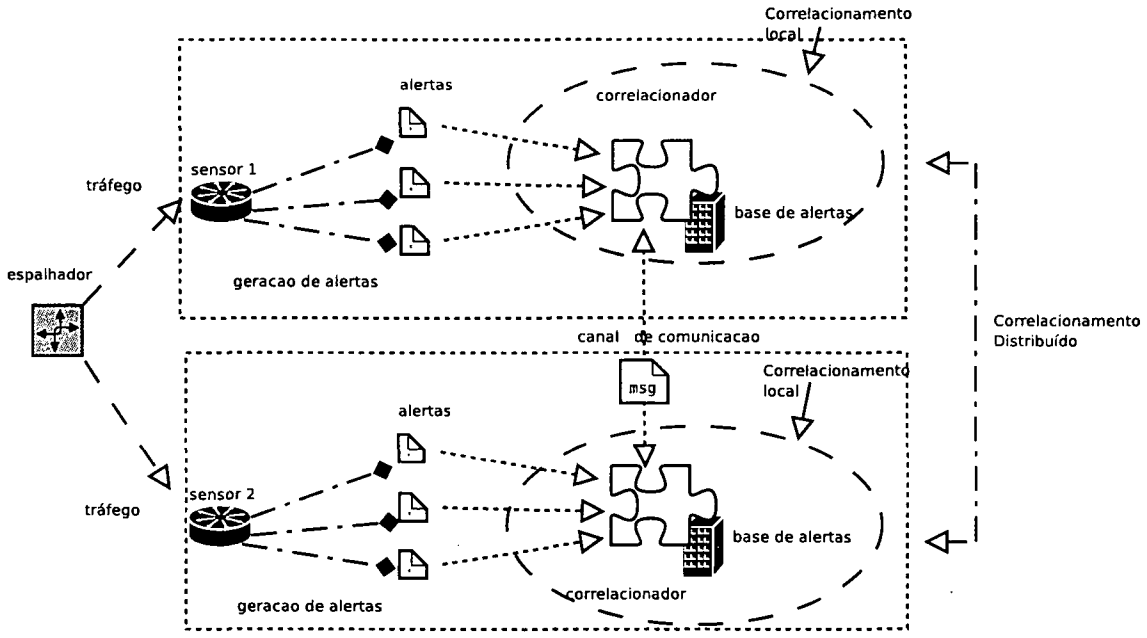

Figura 4.4: Relacionamento entre sensor e correlacionador.

Variáveis de Correlacionamento Os alertas emitidos pelo Snort podem conter as seguintes informações: data, hora, tipo do ataque, protocolo utilizado, endereços IP origem e destino, porta origem e porta destino da tentativa do ataque. Um alerta emitido pelo Snort tem a seguinte aparência[1]:

```
09/11-11:14:26.252905  [**] [1:0:0] WEB-MISC cross site scripting attempt [**]
[Priority: 0] {TCP} 192.168.0.129:47503 -> 192.168.0.95:80
```

As variáveis utilizadas pelo Nariz são: (1) endereço IP origem, (2) endereço IP destino, (3) porta destino, (4) classe do ataque, (5) hora do ataque e (6) data, provenientes das informações geradas pelo alerta do Snort.

As variáveis hora e data do ataque serão utilizadas futuramente com regras de correlacionamento de alertas baseado em tempo. Atualmente, essas variáveis não são usadas em nenhuma regra de correlacionamento. No caso do alerta mostrado acima, os valores das variáveis seriam preenchidos da seguinte forma:

1. endereço IP origem: *192.168.0.95*;
2. endereço IP destino: *192.168.0.129*;
3. porta destino: *80*;
4. classe do ataque: *WEB-MISC*;
5. hora do ataque: *11:14:26*;
6. data: *09/11*.

Além dessas variáveis, outras três variáveis são utilizadas, duas a partir de suposições sobre os endereços de rede origem e destino, removendo o último octeto dos endereços IP origem e destino, respectivamente. A outra variável é o *limiar*, que determina ações a serem executadas, como o correlacionamento distribuído e o envio de mensagens para o administrador da rede. Essas ações são iniciadas através de comparações do valor do *limiar* com valores definidos em um arquivo específico. O *limiar* é uma variável numérica, não negativa, que é incrementada caso um alerta satisfaça a regra de correlacionamento

[1] Alerta emitido pelo Snort com a opção de alerta resumido, opção *-A fast*

com alerta previamente armazenado. O valor inicial da variável *limiar* é zero, essa variável expressa o grau de coincidência entre alertas.

Regras de Correlacionamento As regras de correlacionamento de alertas são baseadas no modelo descrito em [43]. As variáveis descritas acima são agrupadas em uma tabela chamada de *Tec* (Tabela de Endereços e Classe), mostradas na figura 4.5. As regras de correlacionamento consistem de operações binárias para relacionar alertas com características semelhantes.

ipDestino	*ipOrigem*	*portaDestino*	*redeDestino*	*redeOrigem*	*classe*	*hora*	*data*	*limiar*

Figura 4.5: Tabela *Tec*.

Há dois tipos básicos de regras, as *regras de endereço* (*RE*s) e a *regra de classe de ataque* (*RCA*). As *regras de endereço* são formadas por cinco cláusulas, *RE1*, *RE2*, *RE3*, *RE4* e *RE5*, e baseadas nas variáveis *ipDestino*,*ipOrigem*, *portaDestino*, *redeDestino* e *redeOrigem*, respectivamente. Uma cláusula das *REs* é avaliada como verdadeira se o conteúdo de uma das variáveis de um alerta, previamente armazenado, for igual ao conteúdo dessa mesma variável em um novo alerta. A regra *RCA* é formada pela variável *classe*, essa regra é avaliada como verdadeira se o conteúdo da variável *classe* de um alerta já armazenado for igual ao conteúdo dessa mesma variável em um alerta que está sendo avaliado. Para cada avaliação verdadeira das *RE*s e da regra *RCA*, o valor da variável *limiar* é incrementado, e caso um alerta satisfaça todas as regras de correlacionamento, o *limiar* desse alerta será incrementado seis vezes. Toda vez que um alerta é obtido pelo Nariz, são avaliadas as *RE*s e a *RCA*, sendo a variável *limiar* incrementada por qualquer das regras que avalie como verdadeira.

As regras *RE* e *RCA* são consolidadas em uma regra chamada *REC*. A avaliação da regra *REC* é verdadeira se a avaliação da seguinte expressão for verdadeira:

$$REC = ((RE1 \vee RE2 \vee RE3 \vee RE4 \vee RE5) \wedge RCA) \tag{4.1}$$

A regra *REC* é utilizada para a concatenação de alertas, e será descrita adiante. Para eliminar alertas duplicados, o sistema de correlacionamento avalia a chamada *Regra de Duplicata* (*RD*). A avaliação da regra *RD* é verdadeira, se todas as avaliações das *REs* forem verdadeiras e a avaliação da *RCA* for verdadeira:

$$RD = (RE1 \wedge RE2 \wedge RE3 \wedge RE4 \wedge RE5 \wedge RCA) \tag{4.2}$$

As regras de correlacionamento *RD* e *REC* são utilizadas nos correlacionamentos local e distribuído do Nariz.

4.3.3.1 Correlacionamento local

No correlacionamento local, toda vez que um novo alerta é obtido pelo Nariz, é realizada a avaliação das regras *RD* e *REC*. Caso a avaliação das regras *RD* e *REC* retornar falso, o novo alerta é inserido na base dos alertas. Se a avaliação da regra *RD* for verdadeira, então a variável *limiar* do alerta da base que satisfez a regra *RD* é incrementada e o novo alerta é descartado. Se a avaliação da regra *REC* for verdadeira, então a variável *limiar* do alerta também é incrementada e ao invés de descartar o novo alerta é realizada uma "junção" do novo alerta com o alerta pré-existente. Quando é realizado o correlacionamento distribuído, isto é, quando um alerta é obtido através de um Nariz remoto, o incremento do *limiar* pode ser maior, para se diferenciar alertas correlacionados remotamente, que podem ter maior importância (gravidade) que alertas correlacionados localmente. Alertas de vizinhos, de outros Narizes, expressa inúmeros correlacionamentos já detectados pelo vizinho. O correlacionamento distribuído é descrito na seção 4.3.3.2.

Para exemplificar o correlacionamento local e suas regras vamos supor que um ataque do tipo *PortScan* [36] foi detectado por um sensor, que gerou o seguinte alerta:

```
09/03-15:18:29.156466 [**] [117:1:1] (spp_portscan2) Portscan detected from
192.168.1.22: 6 targets 6 ports in 2 seconds [**]
{TCP} 192.168.1.22 -> 192.168.0.3
```

O Nariz obtém o alerta do sensor e tenta correlacionar esse alerta com algum outro alerta de sua base. Supondo que não exista outro alerta na base do Nariz que satisfaça as regras *RD* e *REC*, o Nariz então insere o alerta na base e inicializa com zero o valor da variável *limiar*. Supondo que depois de um minuto o sensor emita outro alerta do mesmo tipo de ataque (*Portscan*) mostrado abaixo:

```
09/03-16:19:22.491698 [**] [117:1:1] (spp_portscan2) Portscan detected from
192.168.1.22: 6 targets 6 ports in 0 seconds [**]
{TCP} 192.168.1.22 -> 192.168.0.3
```

O Nariz extrai esse novo alerta do sensor e tenta correlacionar esse alerta com outro já armazenado em sua base. Para correlacionar os alertas, o Nariz avalia a regra *RD*. Se esta avaliar como falsa, será testada a regra *REC*. No caso, os alertas exemplificados satisfazem a regra *RD*, pois a base de alertas do Nariz já possui um alerta com características idênticas (exceto as varáveis *data* e *hora*.). É incrementada a variável *limiar* e o novo alerta é descartado por ser uma duplicata de um alerta já registrado na base.

Vamos supor que o sensor identificou outro ataque. O ataque identificado é do tipo DDoS (*Distributed Denial of Service*) [39, 18] e o seguinte alerta é emitido pelo sensor:

```
11/30-14:18:16.323155 [**] [1:1855:2] DDOS TFN Probe [**]
[Classification: Attempted Denial of Service]
[Priority: 2] {TCP} 192.168.100.28 -> 61.134.3.11[**]
```

Supondo que não seja encontrado nenhum alerta na base do Nariz que satisfaça as regras *RD* e *REC*, o alerta é então inserido na base, e a variável *limiar* é inicializada com zero. Posteriormente, o sensor identificará outro ataque do tipo DDoS e emite um novo alerta:

```
11/30-14:25:10.321113 [**] [1:1855:2] DDOS TFN Probe [**]
[Classification: Attempted Denial of Service]
[Priority: 2] {TCP} 192.167.10.2 -> 61.134.3.11[**]
```

O Nariz obtém o novo alerta e avalia as duas regras de correlacionamento. A regra avaliada verificação de alertas duplicados (*RD*) falha porque os alertas não têm todas as

variáveis *Tec* iguais. A regra *REC* é então avaliada como verdadeira, pois a base já contém um alerta com pelo menos uma variável com o valor igual ao do novo alerta (variáveis *classe* e *ipDestino*). Como a regra *REC* é satisfeita, a variável *limiar* é incrementada e o novo alerta é unido ao alerta que satisfez a regra *REC*. Duas *RE*s satisfizeram a avaliação nas variáveis *classe* e *ipDestino*, e portanto o *limiar* é incrementado duas vezes. Na união, as variáveis do novo alerta que não satisfizeram as regras de endereço são concatenadas às variáveis do alerta pré-existente e as variáveis *hora do ataque* e *data*, são atualizadas com os valores do novo alerta. Nesse esquema perde-se o histórico de detalhado de hora e data, um esquema que podia ser empregado era guardar essas informações para um histórico detalhado da tentativa. Para o exemplo visto acima dos dois alertas com o mesmo tipo de ataque DDoS, o alerta resultante da união teria o seguinte conteúdo das variáveis:

1. endereço IP origem: *192.168.100.28, 192.168.10.2*;
2. endereço IP destino: *61.134.3.11*;
3. endereço de rede origem: *192.168.100, 192.168.10*;
4. endereço de rede destino: *61.134.3*;
5. porta destino: *NULL*;
6. classe do ataque: *DDOS*;
7. *limiar*: *2*;
8. hora do ataque: *14:25:10*;
9. data: *11/30*.

4.3.3.2 Correlacionamento Distribuído

A função do correlacionamento distribuído de alertas é compartilhar informações sobre possíveis ataques entre correlacionadores e enviar uma mensagem de alerta para o administrador da rede quando necessário. Esse compartilhamento de informações e o envio da mensagem para o administrador da rede é realizado com base no valor da variável *limiar*. Quando um alerta é primeiramente inserido na base do Nariz, o valor do *limiar* é zero. Conforme os alertas forem sendo correlacionados, o *limiar* é incrementado e pode atingir um dos dois valores de gatilho CONVERSA[2] e PANICO[3], que são valores numéricos e não negativos. Toda vez que a variável *limiar* de um alerta é incrementada, seu valor é comparado com esses valores de gatilho, que são determinados previamente. Quando o valor do gatilho CONVERSA é ultrapassado, o Nariz envia mensagens aos outros Narizes, que correlacionam este alerta em suas bases. Quando o valor de PANICO é ultrapassado uma mensagem é enviada ao administrador.

Limiar A definição de valores de gatilho pode ser determinada pelo usuário através de um arquivo chamado *limiar.conf*. Esse arquivo tem a estrutura mostrada na figura 4.6. Na coluna *Classe do Ataque* é definida a classe do ataque, na coluna CONVERSA é definido o valor de *limiar* para o envio de mensagens entre os Narizes, e na coluna PANICO é definido um valor de *limiar* que dispara o envio de mensagens ao administrador da rede.

Classe do Ataque	CONVERSA	PANICO

Figura 4.6: Estrutura do *limiar.conf*.

Se o arquivo de configurações contiver um único registro com [PADRAO, C, P], os valores de C e P serão usados para todos as classes de ataque. O usuário pode definir valores para o gatilho de CONVERSA que provoca a troca de mensagens entre os Narizes, e para o valor de gatilho PANICO que cause o envio de uma mensagem ao administrador. É

[2]Conversa, troca de mensagens entre os Narizes
[3]Pânico, envio de mensagem ao administrador da rede.

possível limitar o correlacionamento a certas classes de alertas, sendo as demais ignoradas.

O valor do *limiar* de um alerta ao longo do tempo é definido pelo número de cláusulas *RCA* e *RE*s avaliadas como verdadeiras, vezes o valor do degrau local (incremento), caso seja um alerta correlacionado localmente; caso seja um alerta obtido remotamente de um vizinho, o valor do *limiar* é definido pelo degrau remoto multiplicado pelo número de cláusulas *RCA* e *RE*s avaliadas como verdadeiras. O degrau remoto é o valor definido para o incremento do *limiar* de um alerta que é correlacionado com um alerta de outro Nariz. O valor do degrau remoto pode ser superior ao incremento local, para aumentar a gravidade de um alerta correlacionado remotamente. A equação 4.4 formaliza o valor do *limiar*.

$$\begin{aligned} limiar[t] &= (\#\text{cláusulas } RCA[t] + \#\text{cláusulas}RE[t]) \times \text{degrau}_{local} \\ &+ \text{alertas}_{vizinhos} \times (\#\text{cláusulas } RCA[t] + \#\text{cláusulas}RE[t]) \times \text{degrau}_{remoto} \end{aligned} \tag{4.3}$$

Para exemplificar a definição dos valores dos gatilhos com relação a tipos de ataques, vamos supor que um usuário somente definiu no arquivo *limiar.conf* a classe de ataque *VIRUS*, com os valores de gatilho CONVERSA e PANICO, como 40 e 70, respectivamente ([VIRUS, 40, 70]). O Nariz descartará todo alerta que não tiver como conteúdo da variável *classe* com valor de *VIRUS*. Quando a quantidade de alertas recebidos da classe *VIRUS* for tal que o *limiar* atinge 40 será enviada uma mensagem em forma de alerta para o(s) Nariz(es) que faz(em) parte do correlacionamento distribuído. Após receber(em) a mensagem, aquele(s) realizará(ão) o correlacionamento local do alerta. Quando o valor do *limiar* chegar a 70 em um dos Narizes, o correlacionador enviará uma mensagem para o administrador da rede. Caso o usuário tivesse acrescentado ao arquivo um registro [PADRAO, 30, 50], o sistema correlacionaria todas as classes de alertas com valores de gatilho CONVERSA e PANICO sendo 30 e 50, exceto os alertas cuja classe de ataque seja de *VIRUS*, que seriam correlacionados com valores de *limiares* definidos para a classe *VIRUS*.

Toda vez que um alerta é correlacionado a variável *limiar* é incrementada. O incremento da variável *limiar* de um alerta que for recebido de outro Nariz pode ser maior que aquele efetuado no correlacionamento local. A figura 4.7 mostra um diagrama de tempos com o comportamento do *limiar* de um alerta armazenado na base. Quando o *limiar* de um alerta é incrementado, o sistema de correlacionamento verifica se o valor do *limiar* desse alerta alcançou um dos dois gatilhos, CONVERSA ou PANICO. A inclinação da reta na figura 4.7 é proporcional ao tamanho do incremento a cada novo correlacionamento. O ponto A da figura mostra que o *limiar* alcançou o gatilho CONVERSA e emite alertas aos Narizes vizinhos. O salto no valor do *limiar* mostrado no ponto B representa o aumento devido ao recebimento de um alerta de outro Nariz. Esse salto é proporcional ao valor do degrau para o correlacionamento distribuído. O Nariz da figura 4.7 envia uma mensagem ao administrador no ponto C, e decrementa o valor do *limiar* fazendo com que o valor do *limiar* fique abaixo do gatilho CONVERSA. O decremento é calculado pela seguinte fórmula: round*(*limiar*/3)*.

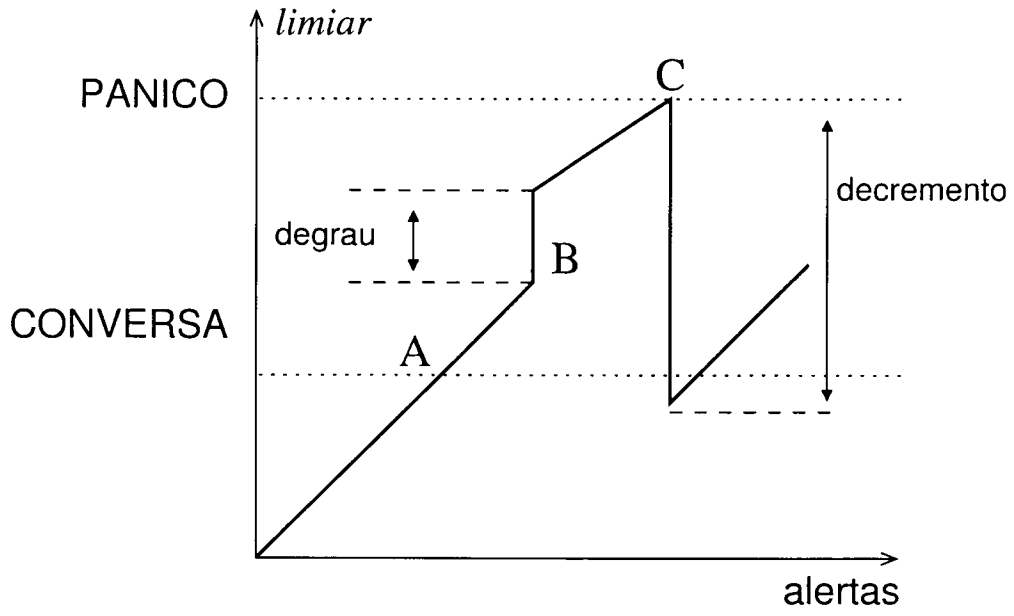

Figura 4.7: Comportamento do *limiar* devido às ações dos gatilhos.

Quando o *limiar* de um alerta ultrapassa o valor de CONVERSA, o Nariz envia este alerta para os outros Narizes. Quando o *limiar* atinge o valor de PANICO uma mensagem é enviada ao administrador da rede e o *limiar* é decrementado. O número mínimo de alertas, com características semelhantes, para enviar mensagem ao administrador é representado pelo valor do gatilho PANICO, que pode ser configurado pelo usuário do Nariz. Se o decremento for pequeno, o *limiar* resultante ficará acima de CONVERSA e o Nariz não emitirá novas mensagens para outros Narizes. A figura 4.8 mostra esta situação, que decorre, ou do pequeno decremento, ou da diferença relativa entre CONVERSA e PANICO.

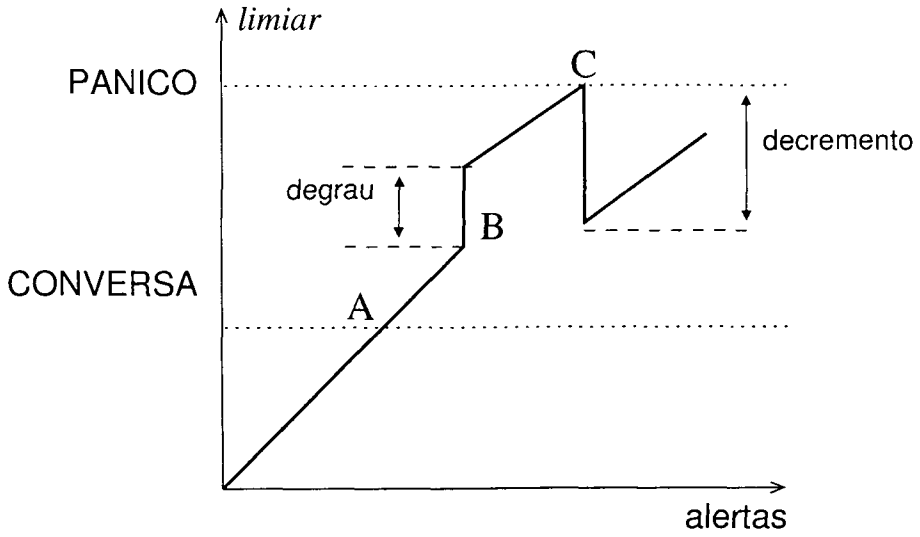

Figura 4.8: Diferença entre valores de gatilho e decremento.

O decremento de round(limiar/3) do *limiar* foi escolhido com base nos resultados de experimentos realizados na fase inicial de desenvolvimento do Nariz. Foram avaliadas cinco fórmulas para o decremento:

$$limiar = panico/10 \tag{4.4}$$

$$limiar = conversa/10 \tag{4.5}$$

$$limiar = \text{round}(\text{limiar}/3) \tag{4.6}$$

$$limiar = (panico - conversa) - degrau \tag{4.7}$$

$$limiar = 0 \tag{4.8}$$

O decremento da equação 4.7 coloca o *limiar* logo abaixo de CONVERSA, causando freqüêntes trocas de mensagens entre os Narizes (na simulação isso ocorreu porque só um Nariz foi simulado). Com os valores de CONVERSA e PANICO utilizados, as equações 4.4 e 4.5 tem comportamento similar porque PANICO/CONVERSA<10. A equação 4.6 produz números relativamente baixos de mensagens ao administrador. A equação 4.8 tem o menor número de mensagens ao administrador, dentre dessas 5 fórmulas, devido ao corte do *limiar* em 0.

Note que a diferença entre os valores de CONVERSA e PANICO é inversamente proporcional ao número de mensagens enviadas ao administrador.

Um outro modelo para o arquivo *limiar.conf* é baseado no seguinte esquema, após a definição para o gatilho PANICO no arquivo, seriam utilizadas mais 8 colunas. Dessas 8 colunas incrementadas, as 6 primeiras seriam utilizados para determinar o valor incremento do *limiar* para cada uma das seis regras de correlacionamento, *REs* e *RCA*. As duas últimas colunas seriam utilizadas para configurar o valor do degrau remoto e identificar uma, de uma possível lista de fórmulas de decremento do *limiar*.

Configuração de Narizes O correlacionamento de alertas tem por finalidade reduzir a quantidade de informação que deve ser processada pelo administrador da rede. Neste sentido, o correlacionamento funciona como um filtro que elimina informação redundante e consolida as informações relevantes. O sistema de correlacionamento Nariz pode ser configurado, permitindo ajustes finos na filtragem da informação. A seguir são discutido três problemas de configurações que devem ser resolvidos quando o sistema é configurado.

Problema 1: Podem ocorrer situações quando uma seqüência de alertas similares sejam distribuídos entre Narizes, mas em nenhum deles é atingido o valor do gatilho CONVERSA, e portanto, o administrador da rede não é avisado da tentativa de ataque. Para evitar que isso ocorra, os valores para os gatilhos de CONVERSA e de PANICO devem ser estabelecidos na proporção inversa ao número de Narizes:

CONVERSA $\propto$ 1/(número de Narizes), PANICO $\propto$ 1/(número de Narizes).

Problema 2: Quanto maior for o degrau de incremento do *limiar*, mais rápido um alerta será emitido ao administrador da rede. O valor do degrau influi diretamente no número de mensagens transmitidas pelos Narizes, fazendo com que o gatilho de PANICO seja atingido rapidamente, causando assim o envio de mensagens repetidas ao administrador. A solução neste caso é escolher um degrau pequeno para o incremento do *limiar* por conta da correlação distribuída, e um valor elevado para o gatilho do PANICO.

Problema 3: As soluções para os Problemas 1 e 2 são conflitantes. O administrador da rede deve definir valores para os gatilhos de CONVERSA e PANICO tais que um número relativamente pequeno de alertas dispare o correlacionamento distribuído. O valor do degrau tem um efeito multiplicativo com relação ao correlacionamento local. Se o degrau for igual a N, cada mensagem recebida de outro Nariz equivale a N alertas correlacionados localmente, um para cada Nariz no sistema. No outro extremo, se o degrau for igual a CONVERSA, cada mensagem recebida de outro Nariz equivale ao número de alertas correlacionados localmente que dispararia a emissão de mensagens aos demais Narizes, compondo um efeito multiplicativo do degrau.

A diferença entre os valores de PANICO e CONVERSA, juntamente com o degrau, determinam o número de mensagens enviadas ao administrador da rede. Quanto maior a diferença de CONVERSA e PANICO e menor o degrau, mais demora para o sistema de correlacionamento enviar mensagens ao administrador.

$$\#msgsAdmin \propto degrau + (\text{PANICO} - \text{CONVERSA}) \tag{4.9}$$

4.3.3.3 Base de Alertas

Para armazenar os alertas é utilizada a biblioteca de banco de dados de domínio público *SQLite* [5]. Essa biblioteca disponibiliza um conjunto de classes para gerenciamento e acesso a dados na base. A *SQLite* é executada no mesmo espaço de endereçamento da aplicação, dispensando a comunicação entre processos ou comunicação através da rede.

Cada base de alertas dos correlacionadores utiliza a biblioteca *SQLite* para servir de repositório de alertas, e a base de alertas é mantida na memória principal para aumentar a velocidade do correlacionamento de alertas.

Desempenho Para inserir e correlacionar 9 mil registros diferentes de alertas na base o espaço alocado foi de 12 Mb. O tempo utilizado para alocar esse espaço em memória secundária (disco rígido) foi de 5 minutos. Já utilizando memória principal o tempo caiu para 15 segundos.

Remoção de Alertas Velhos É necessário utilizar um mecanismo para eliminar alertas considerados "velhos". O mecanismo de remoção de alertas considerados velhos não foi implementado. Quando a base de alerta estiver utilizando 70% da memória principal deve ser executada uma rotina para remover alertas velhos. A rotina executa um "envelhecimento" dos alertas, onde seriam envelhecidos metade do total de alertas em ordem decrescente por data e hora. Esse envelhecimento poderia ser de 3 formas, ou descarregar metade de alertas para a memória secundária, para um base central de alertas, ou simplesmente apagá-los. Uma outra rotina de remoção de alertas, apaga até 50% dos alertas mais velhos e com *limiar* abaixo de CONVERSA.

4.3.3.4 Sobrecarga no Nariz

Foi realizado um experimento para demonstrar a sobrecarrega no sistema de correlacionamento de alertas foi utilizado, como o Nariz, um computador Intel Pentium III de 500 Mhz com dois processadores e com 256 MB de memória RAM. Ao correlacionar 192 mil alertas diferentes em aproximadamente 5 minutos o Nariz não funcionou devido à limitação da memória RAM, pois 192 mil alertas correspondem a 251 MB de memória na base de alertas, que por sua vez pode ser implementada na memória principal. Esse experimento não utilizou nenhum intervalo entre os 192 mil alertas correlacionados.

Para contornar essa sobrecarga 3 soluções podem se empregadas: ou (1) aumentar a memória RAM, ou (2) criar a base de alertas do Nariz na memória secundária ou (3) distribuir esses alertas para mais de um Nariz. A solução de número 3 foi testada e obteve o seguinte resultado. Quando o número de alerta foi divido entre 2 Narizes, cada um correlacionou 96 mil alertas ocupando um espaço de 126 Mb na memória principal. Na divisão de alertas para 3 Narizes, cada Nariz correlacionou 64 mil alertas, onde ocupou 84 Mb em cada base de alertas na memória, portanto ao dividir o total de alerta a mais dois Narizes, o correlacionamento foi realizado com êxito e sem sobrecarregar o sistema.

4.4 Experimentos no Correlacionamento Distribuído

Para avaliar o comportamento de um Nariz, foram realizados experimentos com 500 alertas iguais. O desempenho de cada uma das fórmulas de decremento é medido pelo número de mensagens enviadas ao administrador. O degrau tem valor 3, e quando o limiar de CONVERSA e atingido, o Nariz envia uma mensagem para si próprio. Os valores de CONVERSA e PANICO foram (20, 40), (20, 60) e (20, 80). A figura 4.9 mostra o resultado de simulações com o uso das 5 fórmulas (equação 4.4, 4.5, 4.6, 4.7 e 4.8). Os testes da figura demostram o uso de cada uma das fórmulas associados a valores para os gatilhos CONVERSA e PANICO.

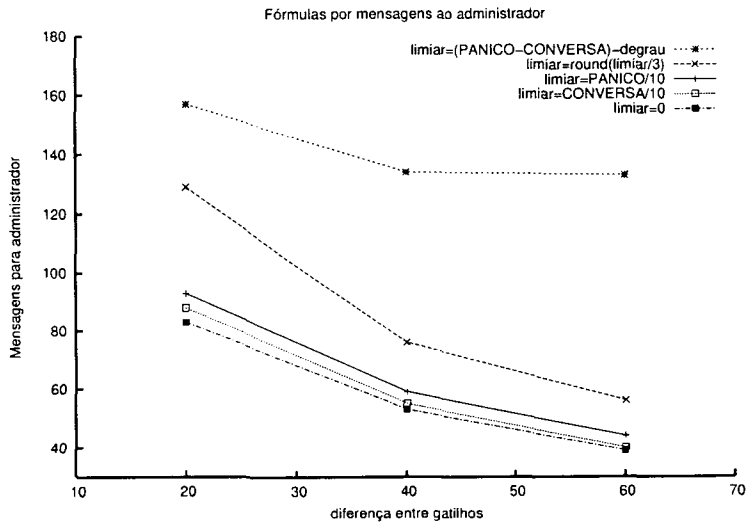

Figura 4.9: Utilização de 5 fórmulas diferentes de decremento no correlacionamento.

A tabela 4.1 contém informações utilizadas para ilustrar a figura 4.9. A primeira coluna representa a fórmula de decremento utilizada, o restante das colunas indicam a quantidade de mensagens emitidas para o administrador em razão das diferenças entre o gatilhos PANICO e CONVERSA.

Fórmula	**20**	**40**	**60**
CONVERSA/10	20	59	44
round(limiar/3)	129	76	56
(PANICO- CONVERSA) - DEGRAU	157	134	133
PANICO/10	88	55	40
0	83	53	39

Tabela 4.1: Dados obtidos de experimentos com 4 fórmulas diferentes de decremento.

Para evidenciar a relação entre os três parâmetros de configuração foram efetuados 3 experimentos, os dois primeiros experimentos correlacionam 100 alertas idênticos e 3 Narizes. Foram correlacionados 100 alertas quando foi utilizado 1 Nariz, 50 para cada um de dois Narizes e 34, 33 e 33 alertas quando se utilizou três Narizes. O valor do degrau foi 1 e a fórmula de decremento round(limiar/3).

No primeiro experimento o valor do gatilho CONVERSA se manteve constante em 40 e foi testado com 6 valores diferentes de gatilho PANICO (50, 60, 70, 80, 90 e 100). A figura 4.10 mostra o efeito de filtragem que o Nariz realiza. Nesse experimento quando foi aumentado o número de Narizes, o número de alertas emitido ao administrador diminuiu.

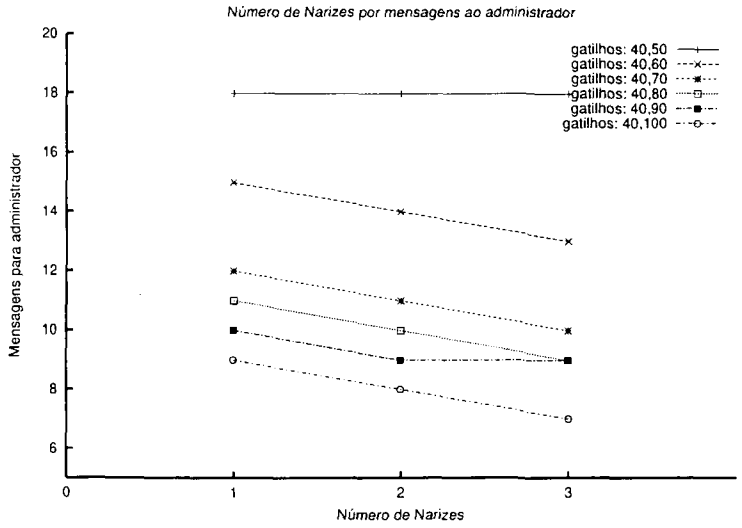

Figura 4.10: Mensagens emitidas ao administrador variando o valor de PANICO.

O gráfico 4.10 ilustra os dados da tabela 4.2 obtidos pelo experimento realizado. A primeira coluna contém os valores de gatilhos de CONVERSA e PANICO, respectivamente. A segunda coluna os alertas emitidos para o administrador da rede quando foi utilizado 1 Nariz, na terceira coluna as mensagens quando foi utilizado 2 Narizes e na quarta coluna contém o número de mensagens quando foi utilizado 3 Narizes.

Gatilhos	**1**	**2**	**3**
40,50	18	18	18
40,60	15	14	13
40,70	12	11	10
40,80	11	10	9
40,90	10	9	9
40,100	9	8	7

Tabela 4.2: Número de mensagens obtidas pelo administrador com o valor de PANICO variando.

No segundo experimento foi testado diferentes valores de gatilho CONVERSA (40, 50, 60, 70, 80 e 90) com PANICO constante em 100. Os resultados obtidos por esse experimento são demonstrados pela figura 4.11, que também evidencia o efeito de filtragem. Os valores obtidos nesse experimento demonstram que para os parâmetros escolhidos (gatilhos, decremento e degrau) os resultados obtidos ficaram bem próximos.

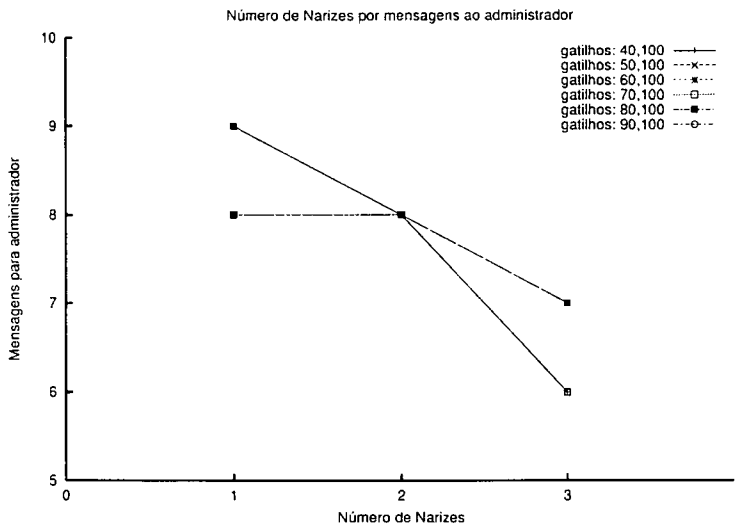

Figura 4.11: Mensagens emitidas ao administrador variando o valor de CONVERSA.

Na tabela 4.3 está contido os dados usados para ilustrar a figura 4.11. Nessa tabela o valor de PANICO foi constante e o de CONVERSA foi variado, como mostra a primeira coluna. As três outras colunas contém dados que representam o número de mensagens recebidas pelo administrador quando se correlacionou alertas para 1, 2 e 3 Narizes.

Gatilhos	1	2	3
40,100	9	8	6
50,100	9	8	7
60,100	8	8	7
70,100	8	8	6
80,100	9	8	7
90,100	8	8	6

Tabela 4.3: Mensagens recebidas administrador com variando o valor de CONVERSA.

No terceiro experimento foram efetuados testes com 500 alertas idênticos e 3 Narizes. Os 500 alertas foram emitidos para um Nariz, 250 alertas para cada um de dois Narizes, e 166, 166, 167 alertas para cada um de três Narizes. Nos três casos os valores dos gatilhos CONVERSA e PANICO (C,P) foram (50,60), (50, 70) e (70,100) e o decremento é round(limiar/3). Em todos estes casos o degrau foi de 3, para alertas recebidos remotamente, os resultados são mostrados na figura 4.12, que demonstra claramente a redução no número de mensagens ao administrador como uma função dos valores dos gatilhos.

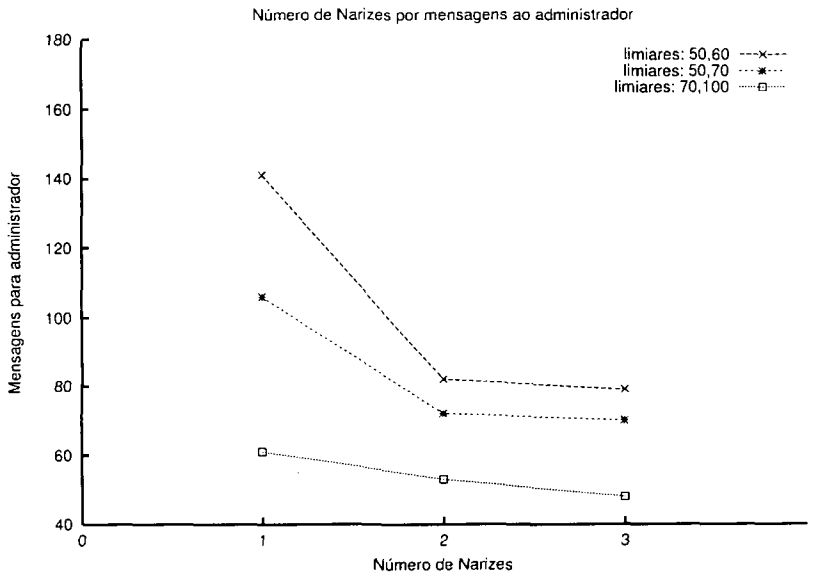

Figura 4.12: Relação entre Narizes, gatilhos e mensagens ao administrador.

Os dados da tabela 4.4 foram utilizados para a figura acima que demonstra a razão entre Narizes, gatilhos e mensagens enviadas ao administrador da rede.

Número de Narizes	Msgs Admin	Gatilhos
1	141	50,60
2	82	50,60
3	79	50,60
1	106	50,70
2	72	50,70
3	70	50,70
1	61	70,100
2	53	70,100
3	48	70,100

Tabela 4.4: Experimentos com 500 alertas iguais variando o número de Narizes.

A queda do número de mensagens para o administrador representa o filtro obtido com o sistema. Com esse filtro, o Nariz, reduz o número de alertas gerados ao administrador da rede.

Capítulo 5

Conclusão

Sistemas de Detecção de Intrusão para Redes são sistemas capazes de detectar e relatar atividades consideradas ofensivas a uma rede. Esses relatos são formados por *alertas*, que possuem informações de atividades consideradas ofensivas. A detecção de intrusão é realizada por análise de tráfego, quanto mais alto o tráfego, mais o Sistema de Detecção de Intrusão para Redes tem necessidade de processamento analisá-lo. Uma alternativa para melhorar o desempenho deste sistema é distribuir o tráfego na rede. Cada módulo desse sistema distribuído é responsável por analisar uma fração do tráfego, e gerar alertas, caso suspeite de alguma atividade ofensiva. Nessa arquitetura pode haver grande quantidade de alertas, por poder relatar os mesmos alertas em mais de um sistema, relatar alertas de forma infundada, ou relatar alertas legítimos. Toda vez que um alerta é relatado, o administrador da rede deve ser capaz de analisá-lo, para realizar alguma ação em *resposta ao incidente*. O correlacionamento de alertas permite reduzir o número de alertas remetidos ao administrador bem como o número de alertas infundados.

Este trabalho apresenta um sistema de correlacionamento distribuído de alertas, chamado *Nariz*. Esse sistema visa correlacionar os alertas relatados pelos Sistemas de Detecção de Intrusão distribuídos, de modo a diminuir o número de mensagem que o administrador da rede deve analisar. Em cada um desses sistemas de detecção é implantado um Nariz para relacionar alertas distribuídos. Esse relacionamento distribuído pode ser realizado em máquinas de menor processamento do que um correlacionamento centralizado, onde

o gargalo de um correlacionamento centralizado é o processamento da máquina.

O correlacionamento de alertas é realizado através de aplicação de regras simples de relacionamento de informações de alertas. O correlacionamento de alertas é executado em duas fases, na primeira fase são aplicadas regras de correlacionamento a alertas locais. A segunda fase é a fase de correlacionamento distribuído, nela é realizada a aplicação das regras a alertas remotos. Os alertas remotos são distribuído entre os Narizes, que fazem parte do sistema, através de troca de mensagens.

Nessas duas fases de correlacionamento são aplicadas as mesmas regras, tanto para alertas locais, como para alertas remotos. As regras fazem comparações com alertas recebidos pelo Nariz com alertas já inseridos na base do Nariz. Se um alerta não for avaliado como verdadeiro por uma das regras de correlacionamento, então este é inserido como um novo alerta na base do Nariz que o recebeu. Toda vez que um alerta é avaliada como verdadeiro é assinalado um incremento em uma variável chamada *limiar* no alerta da base, que satisfez junto ao alerta correlacionado por uma das regras de correlacionamento. Quando *limiar* é incrementado, o valor resultante desse incremento é comparado com dois valores definidos de gatilhos, gatilho CONVERSA e gatilho PANICO. Cada gatilho executa uma função diferente, o gatilho CONVERSA envia alertas para os Narizes do sistema e o gatilho PANICO envia para o administrador. Se o valor da variável *limiar* de um alerta for igual ao gatilho CONVERSA, então é disparado esse alerta em forma de uma mensagem para outros Narizes que fazem parte do sistema de correlacionamento. Se o valor do *limiar* de um alerta for igual ao gatilho PANICO, é enviada uma mensagem do alerta ao administrador da rede. Através desse sistema é diminuído o número de alertas para o administrador, como um efeito de filtro.

Entre as contribuições deste trabalho está a segunda fase de correlacionamento de alertas, o correlacionamento distribuído, que correlaciona alertas remotos. A utilização de gatilhos para o envio de mensagens entre os Narizes e para o administrador, e o arquivo de configuração do Nariz que permite ajustar o correlacionamento, também estão entre as contribuições.

O ajuste dos parâmetros do Nariz é uma tarefa que requer um ajuste fino no seu arquivo de configuração, requer decisões quanto ao número de mensagens a ser emitidas para os Narizes e o administrador, e especificar as classes de ataques que serão correlacionados. O sistema foi testado com 3 número de correlacionadores, e percebeu-se que de um Nariz até três Narizes, os alertas emitidos ao administrador foi diminuindo. O Nariz também foi testado com diferentes valores de gatilhos, evidenciando que com o gatilho PANICO alto e o degrau com valor baixo, o número de mensagens emitidas ao administrador é reduzido.

Trabalhos futuros incluem a definição de regras de correlacionamento baseado em tempo e o envelhecimento nos alertas da base de Narizes. Um novo arquivo de configuração que permita mais ajustes ao correlacionamento, também está incluído em trabalhos futuros.

Referências Bibliográficas

[1] Brazilian Computer Emergency Response Team. www.nbso.nic.br/.

[2] RealSecure SiteProtector Security Fusion Module. www.iss.net/.

[3] Recent advances in intrusion detection. www.raid-symposium.org/raid98/.

[4] Science Applications International Corporation. www.saic.com.

[5] SQLite - An Embeddable SQL Database Engine. www.sqlite.org/.

[6] tcpdump/libcap. www.tcpdump.org.

[7] tcpreplay. tcpreplay.sourceforge.net/.

[8] Intrusion Detection Planning Guide. Technical report, www.cisco.com/univercd/cc/td/doc/product/iaabu/idpg/index.htm, 1999.

[9] Internet Secure Systems & Top Layer, "Gigabit Ethernet Intrusion Detection Solutions". Technical report, Internet Secure Systems & Top Layer Performance Testing, 2000.

[10] Know Your Enemy: Honeynets. Technical report, Honeynets Project, www.honeynet.org, 2003.

[11] J. P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, James P. Anderson Co., Fort Washington, PA, 1980.

[12] Anonymous. *Segurança Máxima para Linux.* Editora Campus, 2a edição edition, 2000.

[13] G. Araribóia. *Inteligência Artificial Um Curso Prático.* Editora Livros Técnico e Científicos, 1988.

[14] R. Bace and P. Mell. Intrusion Detection System. Technical report, NIST - Nation Institute of Standards and Technology, 2001. Special Publication on Intrusion Detection System.

[15] Benjamin Morin, Ludovic Mé, Hervé Debar and Mireille Ducassé. M2D2: A Formal Data Model for IDS Alert Correlation. In *Recent Advances in Intrusion Detection : 5th International Symposium, RAID 2002*, volume 2516, pages 115–137. Springer-Verlag Heidelberg, Outubro 2002.

[16] G. Bruneau. The History and Evolutionof Intrusion Detection. Technical report, SANS Institute, `www.sans.org`, 2001.

[17] M. Burnett. Withstanding Denial of Service Attacks. Technical report, SecurityFocus - Infocus, `www.securityfocus.com`, 2000.

[18] J. B. D. Cabrera, L. L., X. Qin, W. Lee, R. K. Prasanth, B. Ravichandran, and R. K. Mehra. Proactive Detection of Distributed Denial of Service Attacks using MIB Traffic Variables - A Feasibility Study. *7th IFIP/IEEE International Symposium on Integrated Network Management*, 2001.

[19] T. L. Casavant. A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. *IEEE Transactions on Software Engineer*, 14(2), 1988.

[20] T. Champion and M. L. Denz. A Benchmark Evaluation of Network Intrusion Detection Systems. *Aerospace Conference, 2001, IEEE Proceedings*, 6:2705–2712, 2001.

[21] T. Crothers. *Implementing Intrusion Detection Systems: A Hands-On Guide for Securing the Network.* Editora Paperbock, 2002. An Overviwe of Intrusion Detection.

[22] F. Cuppens and A. Miége. Alert Correlation in a Cooperative Intrusion Detection Framework. *Proceeding of the 2002 IEEE Symposium on Security and Privacy (S&P'02)*, pages 187–200, 2002.

[23] D. Curry and H. Debar. Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition. Technical report, IETF: Internet-Draft - 10, `www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-10.txt`, 2001.

[24] et al. D. Pitts, B. Ball. *Red Hat Linux 6, Unleashed.* Editora SAMS, 1999.

[25] R. C. de Souza. Clusters Linux e suas Aplicações. Technical report, Unit - Centro Universitário do Triângulo, Instituto de Ciências Exatas e Tecnológicas, Curso de Ciências da Computação, Uberlândia, 2001.

[26] Hervé Debar and Andreas Wespi. Aggregation and Correlation of Intrusion-Detection Alerts. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 85 – 103. Springer-Verlag, Outubro 2001.

[27] N. Desai. Optimizing NIDS Performance. Technical report, SecurityFocus - Infocus, `www.securityfocus.com`, 2002.

[28] N. Desai. Intrusion Prevetion Systems: the Next Step int the Evolution of IDS. Technical report, SecurityFocus - Infocus, `www.securityfocus.com`, 2003.

[29] Neil Desai. IDS Correlation of VA Data and IDS Alerts. Technical report, SecurityFocus - Infocus, `www.securityfocus.com`, Junho 2003.

[30] D. B. Chapman E. D. Zwicky, S. Cooper and. *Construindo Firewalls Para Internet.* Editora O'Reilly, 2a edição edition, 2000.

[31] S. Seebass T. R. Hein E. Nemeth, G. Snyder. *Segurança Manual de Administração do Sistema Unix.* Editora Bookman, 3a edição edition, 2002.

[32] S. Edwards. Vulnerabilities of Network Intrusion Detection Systems: Realizing and Overcoming the Risks - The Case for Flow Mirroring. Technical report, Top Layer, `cnscenter.future.co.kr/resource/security/ids/IDSB_White_Papera.pdf`, 2002.

[33] N. Einwetcher. Implemeting Networks Taps with Intrusion Detection Systems. Technical report, SecurityFocus - Infocus, `www.securityfocus.com`, 2002.

[34] Joel Eriksson. Dropbear SSH Server Format String Vulnerability. Technical report, SecurityTeam.com - Beyond Security, `www.securiteam.com/unixfocus`, 2003.

[35] G. A. Fink, B. L. Chappell, T. G. Turner, and K. F. O'Donoghue. A Metric-Based Approach to Intrusion Detection System Evaluation for Distributed Real-Time Systems. *Proceedings of the International Parallel And Distributed Processing Symposium (IPDPS'02)*, pages 93–100, 2002.

[36] Flyodor. The Art of Scanning. Technical report, Phrack, `www.phrack.com`, 1997.

[37] The Internet Engineering Task Force. Internet Society. `wwww.ietf.org`.

[38] S. Garfinkel and G. Spafford. *Practical Unix & Internet Security.* Editora O'Reilly, 2a edição edition, 1996.

[39] S. Gibson. Distributed Reflection Denial of Service. Technical report, Gibson Research Corporation, `grc.com/dos/drdos.htm`, 2002.

[40] R. Graham. FAQ: Network Intrusion Detection Systems. Technical report, Rober Graham, `www.it-sec.de/mirrors/ids/idsfaq.html`, 2000.

[41] B. Gruschke. Integrated Event Management: Event Correlation using Dependency Graphs.

[42] L. Heberlein. A network security monitor. *Proceedings IEEE Computer Society Symposium - Research in Security and Privacy*, pages 296–303, 1990.

[43] L. Tinnel J. Haines, D. K. Ryder and S. Taylor. Validation of Sensor Alert Correlators. *IEEE Security & Privacy Magazine*, 1:46–56, 2003.

[44] E. J. Kamerling. The Hping2 Idle Host Scan. Technical report, SANS Institute, `www.sans.org/rr/audit/hping2.php`, 2001.

[45] R. Kay. Event Correlation. Technical report, Computerworld, `www.computerworld.com`, 2003.

[46] R Kemmerer and G Vigna. Intrusion Detection: A Brief History and Overview. *Security & Privacy, Supplement to IEEE Computer Magazine*, pages 27–30, Abril 2002. `www.computer.org/computer/sp/articles/kem/`.

[47] P. Kothari. Intrusion Detection Interoperability and Standardization. Technical report, SANS Institute, `www.sans.org`, 2002.

[48] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer. Stateful Intrusion Detection for High-Speed Networks. *IEEE Symposium on Security and Privacy*, 2002.

[49] B. Laing. How To Guide- Implemeting a Network Based Intrusion Detection System. Technical report, ISS - Internet Security Systems, 2000.

[50] M. Roesch and C. Green. *Snort Users Manual.* `www.snort.org/docs/writing_rules/`, 2001. Snort Release: 1.9.1.

[51] S. Bretz L. R. Pugh K. Suzuki D. H. Wood M. W. Murhammer, O. Atakan. *TCP/IP Tutorial and Technical Overview.* Editora Prentice Hall PTR, 1998.

[52] J. Marsh. XML Base. Technical report, W3C Candidate Recommendation, `www.w3.org/TR/xmlbase`, 2001.

[53] M. Singhal N. G. Shivarati, P. Krueger. Load Distributing for Locally Distributed Systems. *IEEE Computer Magazine*, 25(12), 1992.

[54] S. Northcutt. *Intrusion Detection: Shadow Style - A Primer for Intrusion Detection Analysis Step By Step Guide.* Editora SANS, 2a edição edition, 2000.

[55] Massachusetts Institute of Technology. Lincoln Laboratory. `www.ll.mit.edu/`.

[56] Linux Online. What is Linux. `www.linux.org/info/index.html`.

[57] Windows Family Home Page. Microsoft Windows. `www.microsoft.com/windows/`.

[58] RealSecure Network Protection. Iss - internet security systems. `www.iss.net/products_services/enterprise_protection/rsnetwork/index.ph%p`.

[59] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *USENIX LISA conference*, Novembro 1999.

[60] C. Gary Rommel. The Probability of Load balacing Success in a Homogenous Network. *IEEE Transactions on Software Engineering*, 17(9), 1991.

[61] J. A. Hoagland S. Staniford and J. M. McAlerney. Practical Automated Detection of Stealthy Portscans. *Journal of Computer Security*, 10:105–136, 2002.

[62] Billy Smith. Thinking about Security Monitoring and Event Correlation. Technical report, SecurityFocus - Infocus, `www.securityfocus.com`, Novembro 2000.

[63] L. Spitzner. The Value of Honeypots, Part One: Honeypot Solutions and Legal Issues. Technical report, SecurityFocus - Infocus, `www.securityfocus.com`, 2001.

[64] Stalker. Technical report, Haystack Labs, Inc. `www.haystack.com`.

[65] A. Sundaram. An Introduction to Intrusion Detection. *ACM Crossroads - Student Magazine*, 1996.

[66] SystemWatch and NerveCenter. Security Event Correlation with Open's Security Solutions. Technical report, OpenService, Inc. White Paper, 2002.

[67] A. S. Tanenbaum. *Computer Networks*. Editora Prentice Hall PTR, 4a edição edition, 2002.

[68] R. Vaarandi. SEC -a Lightweight Event Correlation Tool. *IEEE 2002 Workshop on IP Operations and Management*, pages 111–115, 2002.

[69] Alfonso Valdes and Keith Skinner. Probabilistic Alert Correlation. In *Recent Advances in Intrusion Detection (RAID 2001)*, number 2212 in Lecture Notes in Computer Science. Springer-Verlag, 2001.

[70] Y. Wang. Load Sharing in Distributed Systems. *IEEE Transactions on Computers*, c-34(3), 1985.

[71] Sourav Bhattacharya Y. Peggy Shen, Wei-Tek Tsai and Ting Liu. Attack Tolerant Enhancement of Intrusion Detection Systems. *MILCOM 2000. 21st Century Military Communications Conference Proceedings*, 1:425–429, 2000.

Apêndice A

Algoritmo do Sistema de Correlacionamento Distribuído Nariz

Este apêndice contém o algoritmo usado no sistema de correlacionamento desenvolvido neste trabalho, juntamente com seus devidos comentários.

```
// INICIALIZAÇÕES DE VARIÁVEIS


//Variáveis booleanas para checagem de informações do alerta corrente
// com o alerta que está contido na base, campos do alerta: IP Origem,
// IP Destino, Rede Origem, Rede Destino, Porta Destino e Classe,
// respectivamente. Todos inicializados com falso.

booleana IPO, IPD, RO, RD, PD, C;


//Variaveis booleanas para saber se as pesquisas por determinado campo
// foi bem sucedida se todas nao foram bem sucedida, alerta novo. Todos
// inicializados com falso.

booleana pesquisaClasse, pesquisaIpOrigem, pesquisaIpDestino, pesquisaRedeOrigem,
        pesquisaRedeDestino. pesquisaPortaDestino;


//Valores de PANICO e CONVERSA é obtido do limiar.conf
PANdvICO=limiar.conf(PANICO);
CONVERSA=limiar.conf(CONVERSA);



// REGRAS DE CORRELACIONAMENTO


// Verifica se tem alerta no arquivo de alerta do Snort
Enquanto temAlertas(alert) faca
{


  //Correlacionamento de classe do ataque
```

```
//alertaCorrente é o alerta analisado no momento
Se (PesquisaBD(alertaCorrente.classe, alertasBD.classe) == OK)
{

  //Não se trata de um alerta novo
  pesquisaClasse = verdadeiro;

  //Se entrou aqui essa variável é verdadeiro
  C = verdadeiro;

  //para cada alerta que a pesquisa foi bem sucedida
  Para cada alerta faca
  {

    Se (alertaBD[i].limiar = PANICO - 1)
       envia(alerta, administrador);
    Se Nao
       Se (alertaBD[i].limiar = CONVERSA - 1)
   envia(alerta, listaNarizes);


    //alertaBD[i] é um vetor de alerta que retornou da pesquisa
    alertaBD[i].limiar = limiar + 1;


    Se( pesquisa(alertaBD[i].iporigem, alertaCorrente.iporigem) )
       IPO = verdadeiro;

    Se( pesquisa(alertaBD[i].ipdestino, alertaCorrente.ipdestino) )
       IPD = verdadeiro;

    Se( pesquisa(alertaBD[i].redeorigem, alertaCorrente.redeorigem) )
       RDO = verdadeiro;

    Se( pesquisa(alertaBD[i].rededestino, alertaCorrente.rededestino) )
       RD = verdadeiro;

    Se( pesquisa(alertaBD[i].portadestino, alertaCorrente.portadestino) )
       PD = verdadeiro;


    //atualiza as informações diferente do alerta do banco
    //atualiza também hora e data do alerta do banco
    escolheUpdate(IPO, IPD, RO, RD, PD, C);

    proximo(alertaBD[i]);


  }
```

```
}


//Correlacionamento de IP origem do ataque

//alertaCorrente é o alerta analisado no momento
Se (PesquisaBD(alertaCorrente.iporigem, alertasBD.iporigem) == OK)
{

  //Não se trata de um alerta novo
  pesquisaIpOrigem = verdadeiro;

  //Se entrou aqui essa variável é verdadeiro
  IPO = verdadeiro

  //para cada alerta que a pesquisa foi bem sucedida
  Para cada alerta faca
  {

    Se (alertaBD[i].limiar = PANICO - 1)
       envia(alerta, administrador);
    Se Nao
       Se (alertaBD[i].limiar = CONVERSA - 1)
   envia(alerta, listaNarizes);


    //alertaBD[i] é um vetor de alerta que retornou da pesquisa
    alertaBD[i].limiar = limiar + 1;

    Se( pesquisa(alertaBD[i].classe, alertaCorrente.classe) )
       C = verdadeiro;

    Se( pesquisa(alertaBD[i].ipdestino, alertaCorrente.ipdestino) )
       IPD = verdadeiro;

    Se( pesquisa(alertaBD[i].redeorigem, alertaCorrente.redeorigem) )
       RDO = verdadeiro;

    Se( pesquisa(alertaBD[i].rededestino, alertaCorrente.rededestino) )
       RD = verdadeiro;

    Se( pesquisa(alertaBD[i].portadestino, alertaCorrente.portadestino) )
       PD = verdadeiro;


    //atualiza as informações diferente do alerta do banco
    //atualiza também hora e data do alerta do banco
    escolheUpdate(IPO, IPD, RO, RD, PD, C);


    proximo(alertaBD[i]);
```

```
  }

}


//Correlacionamento de IP destino do ataque

//alertaCorrente é o alerta analisado no momento
Se (PesquisaBD(alertaCorrente.ipdestino, alertasBD.ipdestino) == OK)
{

  //Não se trata de um alerta novo
  pesquisaIpDestino = verdadeiro;

  //Se entrou aqui essa variável é verdadeiro
  IPD = verdadeiro;

  //para cada alerta que a pesquisa foi bem sucedida
  Para cada alerta faca
  {

    Se (alertaBD[i].limiar = PANICO - 1)
       envia(alerta, administrador);
    Se Nao
       Se (alertaBD[i].limiar = CONVERSA - 1)
   envia(alerta, listaNarizes);


    //alertaBD[i] é um vetor de alerta que retornou da pesquisa
    alertaBD[i].limiar = limiar + 1;


    Se( pesquisa(alertaBD[i].iporigem, alertaCorrente.iporigem) )
       IPO = verdadeiro;

    Se( pesquisa(alertaBD[i].classe, alertaCorrente.classe) )
       IPD = verdadeiro;

    Se( pesquisa(alertaBD[i].redeorigem, alertaCorrente.redeorigem) )
       RDO = verdadeiro;

    Se( pesquisa(alertaBD[i].rededestino, alertaCorrente.rededestino) )
       RD = verdadeiro;

    Se( pesquisa(alertaBD[i].portadestino, alertaCorrente.portadestino) )
       PD = verdadeiro;

    //atualiza as informações diferente do alerta do banco
    //atualiza também hora e data do alerta do banco
    escolheUpdate(IPO, IPD, RO, RD, PD, C);
```

```
    proximo(alertaBD[i]);
  }

}


//Correlacionamento de rede origem do ataque

//alertaCorrente é o alerta analisado no momento
Se (PesquisaBD(alertaCorrente.redeorigem, alertasBD.redeorigem) == OK)
{

  //Não se trata de um alerta novo
  pesquisaRedeOrigem = verdadeiro;

  //Se entrou aqui essa variável é verdadeiro
  RO = verdadeiro;

  //para cada alerta que a pesquisa foi bem sucedida
  Para cada alerta faca
  {

    Se (alertaBD[i].limiar = PANICO - 1)
       envia(alerta, administrador);
    Se Nao
       Se (alertaBD[i].limiar = CONVERSA - 1)
   envia(alerta, listaNarizes);


    //alertaBD[i] é um vetor de alerta que retornou da pesquisa
    alertaBD[i].limiar = limiar + 1;


    Se( pesquisa(alertaBD[i].iporigem, alertaCorrente.iporigem) )
       IPO = verdadeiro;

    Se( pesquisa(alertaBD[i].ipdestino, alertaCorrente.ipdestino) )
       IPD = verdadeiro;

    Se( pesquisa(alertaBD[i].classe, alertaCorrente.classe) )
       RDO = verdadeiro;

    Se( pesquisa(alertaBD[i].rededestino, alertaCorrente.rededestino) )
       RD = verdadeiro;

    Se( pesquisa(alertaBD[i].portadestino, alertaCorrente.portadestino) )
       PD = verdadeiro;

    //atualiza as informações diferente do alerta do banco
    //atualiza também hora e data do alerta do banco
    escolheUpdate(IPO, IPD, RO, RD, PD, C);
```

```
    proximo(alertaBD[i]);
  }

}


//Correlacionamento de rede destino do ataque

//alertaCorrente é o alerta analisado no momento
Se (PesquisaBD(alertaCorrente.rededestino, alertasBD.rededestino) == OK)
{

  //Não se trata de um alerta novo
  pesquisaRedeDestino = verdadeiro;

  //Se entrou aqui essa variável é verdadeiro
  RD = verdadeiro;

  //para cada alerta que a pesquisa foi bem sucedida
  Para cada alerta faca
  {

    Se (alertaBD[i].limiar = PANICO - 1)
       envia(alerta, administrador);
    Se Nao
       Se (alertaBD[i].limiar = CONVERSA - 1)
   envia(alerta, listaNarizes);


    //alertaBD[i] é um vetor de alerta que retornou da pesquisa
    alertaBD[i].limiar = limiar + 1;


    Se( pesquisa(alertaBD[i].iporigem, alertaCorrente.iporigem) )
       IPO = verdadeiro;

    Se( pesquisa(alertaBD[i].ipdestino, alertaCorrente.ipdestino) )
       IPD = verdadeiro;

    Se( pesquisa(alertaBD[i].redeorigem, alertaCorrente.redeorigem) )
       RDO = verdadeiro;

    Se( pesquisa(alertaBD[i].classe, alertaCorrente.classe) )
       RD = verdadeiro;

    Se( pesquisa(alertaBD[i].portadestino, alertaCorrente.portadestino) )
       PD = verdadeiro;

    //atualiza as informações diferente do alerta do banco
    //atualiza também hora e data do alerta do banco
```

```
    escolheUpdate(IPO, IPD, RO, RD, PD, C);

    proximo(alertaBD[i]);
  }

}


//Correlacionamento de porta destino do ataque

//alertaCorrente é o alerta analisado no momento
Se (PesquisaBD(alertaCorrente.portadestino, alertasBD.portadestino) == OK)
{

  //Não se trata de um alerta novo
  pesquisaPortaDestino = verdadeiro;

  //Se entrou aqui essa variável é verdadeiro
  PD = verdadeiro;

  //para cada alerta que a pesquisa foi bem sucedida
  Para cada alerta faca
  {

    Se (alertaBD[i].limiar = PANICO - 1)
       envia(alerta, administrador);
    Se Nao
       Se (alertaBD[i].limiar = CONVERSA - 1)
   envia(alerta, listaNarizes);


    //alertaBD[i] é um vetor de alerta que retornou da pesquisa
    alertaBD[i].limiar = limiar + 1;


    Se( pesquisa(alertaBD[i].iporigem, alertaCorrente.iporigem) )
       IPO = verdadeiro;

    Se( pesquisa(alertaBD[i].ipdestino, alertaCorrente.ipdestino) )
       IPD = verdadeiro;

    Se( pesquisa(alertaBD[i].redeorigem, alertaCorrente.redeorigem) )
       RDO = verdadeiro;

    Se( pesquisa(alertaBD[i].rededestino, alertaCorrente.rededestino) )
       RD = verdadeiro;

    Se( pesquisa(alertaBD[i].classe, alertaCorrente.classe) )
       PD = verdadeiro;
```

```
      //atualiza as informações diferente do alerta do banco
      //atualiza também hora e data do alerta do banco
      escolheUpdate(IPO, IPD, RO, RD, PD, C);

     proximo(alertaBD[i]);
    }

  }

  // NOVO ALERTA ?

  //Se tudo der falso, trata-se de um novo alerta
  Se( (pesquisaClasse, pesquisaIpOrigem, pesquisaIpDestino, pesquisaRedeOrigem,
        pesquisaRedeDestino. pesquisaPortaDestino) == falso )
insereBD(banco, alertaCorrente);


}
```